МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт энергетики и электротехники
(институт)
Кафедра Промышленная электроника

11.04.04 – Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Электронные приборы и устройства
(направленность (профиль))

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему «Пульт управления для CNC станка»

Студент С.А. Тарасов _____
(И.О. Фамилия) (личная подпись)

Руководитель М.В. Позднов _____
(И.О. Фамилия) (личная подпись)

Консультанты _____ _____
(И.О. Фамилия) (личная подпись)

_____ _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д.т.н., профессор В.В.Ивашин _____
(ученая степень, звание, И. О.Ф.) (личная подпись)

Допустить к защите

Заведующий кафедрой

“Промышленная электроника” к.т.н, доцент А.А. Шевцов _____
(ученая степень, звание, И.О.Ф.) (личная подпись)

«_____»_____2016 г.

Тольятти 2016

АННОТАЦИЯ

Данная магистерская работа посвящена разработке и изготовлению станка с числовым программным управлением и состоит из пояснительной записки объемом 101 листов и графической части, объем составляет 8 листов формата А1.

Целью работы является разработка пульта управления для станка с числовым программным управлением и повышение энергетических характеристик данного проекта.

Задачами работы являются исследование принцип работы станка с числовым программным обеспечением, разработать автономный пульт для управления и обосновать выбор элементов, конструкторский расчет.

В первом разделе рассматривается состояние вопроса и приводятся общие сведения об существующих конструкторских решениях станков с ЧПУ, а также анализ выбора микроконтроллеров.

Во втором разделе производиться анализ принципа работы станков с числовым программным управлением и разработка программы для управления.

В третьем разделе конструкторско-технологические методы реализации сборочных компонентов устройства.

В четвертом разделе производится технико-экономическое обоснование разработки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Описание станка ЧПУ

В настоящее время огромное количество промышленных предприятий, в чью сферу деятельности входит механическая обработка, располагают огромным парком станков, имеющие системы числового программного управления (*ЧПУ*). Эти станки в предназначены для выполнения тех же производственных задач, что и их аналоги с ручным или механическим управлением. Разница между ними лишь в том, что перемещение рабочих органов станков с *ЧПУ* осуществляется за счет электроники, под управлением специальной компьютерной программы[1].

Рис.1 ЧПУ станок

Преимущества использования станков с числовым программным управлением в том, чтобы обеспечении более качественный уровень автоматизации производственного процесса. Изготовление деталей ведется в автоматическом режиме, практически без участия человека, оператора-станочника, роль которого заключается в выполнении операций контроля за процессом

и участии на подготовительном и завершающем этапе: в первоначальной наладке и последующему контролю за ходом выполнения программы и соблюдением автоматикой всех технологических процессов. Автономная работа станков с *ЧПУ* может продолжаться беспрерывно и достаточно долго, причем качество получаемого изделия остается высоким. Таким образом, один человек может одновременно следить и контролировать большой станочный парк с *ЧПУ*.

Еще одно значительное преимущество – обеспечение производственной универсальности: чтобы станок перешел к изготовлению другого вида продукции, необходимо лишь заменить специально подготовленное программное обеспечение. При этом в любой время можно вернуться к предыдущей программе, уже проверенной в деле. Замену управляющих программ можно осуществлять бесчисленное множество.

Третье немаловажное преимущество – наличие возможности повторять одни и те же действия многократно. Одна и та же программа позволяет производить на требуемом уровне качества тысячи полностью одинаковых по отношению друг к другу деталей. К тому же использование ЧПУ позволяет рабочему механизму выполнять максимально малое количество действий для получения необходимого результата. Как следствие мы получаем максимальную загрузку оборудования и получения максимальной производительности.

Наконец, четвертое преимущество– возможность изготавливать детали достаточно сложной формы, такие, изготовление которых с использованием обычной станочной техники не представляется возможным[2].

Автоматическое числовое программное управление станком осуществляет промышленный компьютер – важная составляющая станка – при помощи специального программного обеспечения. В программу заложены все параметры обработки и траектории рабочего механизма позволяющие четко соблюдать режимы резания и осуществлять полноценную обработку. Изобретение и последующее совершенствование станков с ЧПУ, без преувеличений, явилось наступлением нового этапа научно-технической революции – ведь ранее управлять станками приходилось исключительно в ручном режимах.

Считывая данные управляющего программного обеспечения, компьютерный модуль посылает на соответствующие двигатели те или иные электронные сигналы. А значит, фактически управляет всеми циклами обработки, поскольку под воздействием двигателей происходит перемещение непосредственно осуществляющих механическую обработку детали рабочими механизмами станка[3].

1.1 Обоснование и выбор типа управления станком с ЧПУ

Исходя из информационного признака и по количеству потоков информации для управления станком с числовым программным управлением используется замкнутый и разомкнутый тип. Рисунок 2.

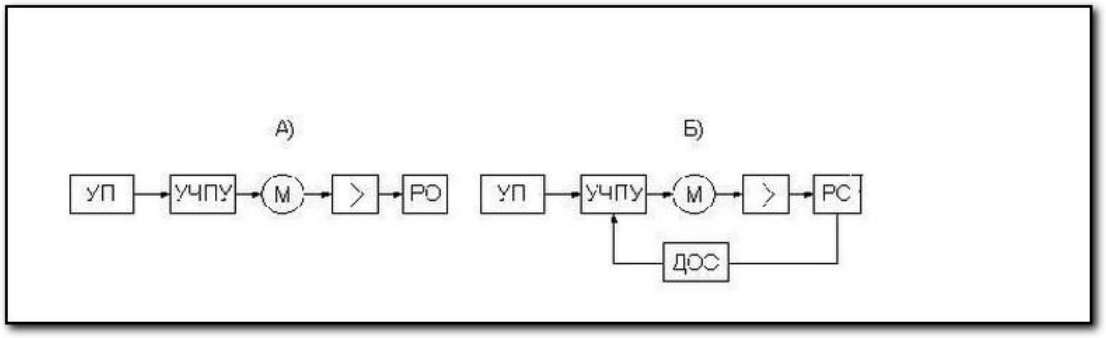

Рисунок 2 - типы управления станком ЧПУ а) разомкнутое, б) замкнутое

В случае разомкнутого управления существует только один информационный поток, который направляется от УП (управляющий поток) на РО (рабочий орган). Станки, имеющие такое управление, оснащены шаговыми двигателями ШД, мотором М.

В случае замкнутого управления существует два информационных потока: один идет от УП, другой — от ДОС (датчиков обратной связи).

Конструктивно наиболее сложными являются замкнутые системы управления, однако они и функционируют точнее последних, поскольку в них фактическая отработка передвижений сравнивается с заданной, а также выполняется коррекция УЧПУ по информации датчиков обратной связи.

В системах управления замкнутого типа выделяются адаптивные или самонастраивающиеся системы. Такие системы содержат дополнительные информационные потоки, способные корректировать процедуру обработки в зависимости от реальных условий (колебания припуска, затупление инструмента и пр.).

В современных ЧТУ станках как правило в основном используются шаговые двигатели. Благодаря практически полной безинерционности

шаговых двигателей и огромного крутящего момента, появилась возможность полностью отказаться от датчиков обратной связи, что существенно упрощает конструкцию станка и его стоимость[4].

Управление шаговым двигателем

Шаговый электродвигатель (ШД) - это синхронный бесщеточный электродвигатель с несколькими обмотками, в котором ток, подаваемый в одну из обмоток статора, вызывает фиксацию ротора. Для его работы требуется специальное электронное устройство – драйвер управления (ДУ) ШД. Современные драйверы работают по определенному протоколу, с регулируемыми уровнями входных сигналов, требуемыми логическими уровнями. Большинство ДУ работает по протоколу STEP/DIR/ENABLE и совместимы с TTL уровнем сигналов 0..5 В[5].

Драйвера управления шаговым двигателем ЧПУ станка выбирают по следующим параметрам:

- Сила тока. Ток, который может обеспечить ДУ. Как правило, он изменяется в достаточно большом пределе, но ДУ нужно подбирать такой, который сможет выдавать ток, равный току фазы данного ШД. Желательно, чтобы максимальная сила тока ДУ была на 20 - 30% больше. Это дает запас на случай, если необходимо получить несколько больший момент от ШД. Производители периодически выпускают пары ШД/ДУ с оптимально выбранными характеристиками. Пример HY-TB3DV-M / NEMA23HS7430.
- Напряжение питания. Его значение достаточно сильно влияет на момент силы и скорость переключения двигателя, а так же на вибрации, нагрев ШД. Как правило, максимальное питающее напряжение ДУ примерно равно предельному току I, помноженному на 7-10. Чем выше индуктивность ШД - тем выше напряжение необходимое для драйвера. Имеется эмпирическая формула $U = 32\sqrt{L}$, где L - индуктивность обмотки ШД. U - максимальное значение напряжения питания драйвера. Значение U, получаемая по представленной формуле, приблизительная, но это значение

позволяет отталкиваться при выборе ДУ.

- Опторазвязка по входу. Практически во всех ДУ, опторазвязка стоит обязательно т.к. пробой ключа может привести к мощному импульсу, который выведет из строя дорогостоящий ЧПУ- контроллер. Опторазвязка дополнительно уменьшает наведение помех от силовой части ДУ в управляющий контроллер.
- Функция подавляющая резонанса. Резонанс ШД – явление, которое появляется постоянно и отличается только резонансной частотой. Она, в первую очередь, зависит от момента инерции нагрузки, питающего напряжения ДУ и заданной силы тока фазы ШД. При появлении резонанса ШД начинает вибрировать и снижать крутящий момент, вплоть до остановки вала. Для уменьшения резонанса применяют микрошаг и – встроенные алгоритмы подавления резонанса. Вибрирующие в резонансе ротор ШД создает микроколебания ЭДС индукции в обмотках, и по их параметру и амплитуде ДУ определяет, наличие резонанса и его величину. В зависимости от полученных значений ДУ незначительно сдвигает шаги двигателя по времени относительно друг друга – такая созданная неравномерность компенсирует резонанс.
- Протокол управления. Необходимо убедиться, что ДУ работает по необходимому вам протоколу, а уровни входящих сигналов соответствуют с требуемыми нам логическими уровнями. Большинство ДУ общаются по протоколу STEP,DIR,ENABLE и совместимо с TTL уровнем сигналов 0..5 В.
- Присутствие защитных функций. Среди которых защита от превышения напряжения питания, тока обмоток, в т.ч. от короткого замыкания обмоток, от переполюсовки напряжения питания, от неправильного подключения фаз ШД.

1.2 Анализ схемотехнических решений станков с ЧПУ:

Общая структура станков с ЧПУ:

- **пульт оператора (или консоль ввода-вывода)**, дающий возможность вводить управляющую программу, устанавливать режимы работы; сделать операцию вручную. Обычно, внутри шкафа пульта управления современной ЧПУ расположены её остальные части;
- **дисплей (или операторская панель)** — для зрительного контроля режимов работы и изменяемой управляющей программы/данных; может быть выполнен в виде отдельного оборудования для дистанционного контроля оборудованием;

- **контроллер** — компьютеризированное оборудование, выполняющий задачи формирования траектории перемещения рабочего инструмента, технологических команд управления средствами автоматики оборудования, общим управлением, изменением управляющих программ, диагностики и дополнительных расчетов (траектории движения рабочего инструмента, режимов работы);
- **ПЗУ** — память, необходимая для долгосрочного хранения (годы и десятки лет) рабочих программ и констант; информация из ПЗУ доступна только для считывания;
- **ОЗУ** — память, предназначенная для краткосрочного хранения рабочих и системных программ, используемых в текущий момент[6].

В роли **контроллера** выступает промышленный контроллер: микропроцессор, на котором построена встраиваемая система; программируемый логический контроллер или значительно усложненное устройство управления, такое как промышленный компьютер.

Важным параметром CNC-контроллера является количество управляемых осей, которые он способен синхронизировать для

одновременного перемещения — этого требует высокую производительность и необходимое ПО.

В качестве применяемых механизмов используются сервоприводы, ШД.

Для передачи информации между исполнительным оборудованием и системой управления станка, как правило, используется промышленная сеть. В более простых станках шкаф управления монтируется в непосредственной близости от исполнительных устройств.

Управляющая система считывает инструкции специализированного языка программирования программы, который затем интерпретатором системы ЧПУ переводится из входного языка в команды управления главным приводом, приводами подач, контроллерами управления узлов станка (например, включить/выключить подачу охлаждающей эмульсии).

Разработка управляющих программ в настоящее время выполняется с использованием специальных модулей для систем автоматизированного проектирования (САПР) или отдельных систем автоматизированного программирования (CAM), которые по электронной модели генерируют программу обработки.

Наиболее распространенный язык программирования ЧПУ для металлорежущего оборудования описан документом ISO 6983 Международного комитета по стандартам и называется «G-код». В отдельных случаях — например, системы управления гравировальными станками — язык управления принципиально отличается от стандарта. Для простых задач, например, раскроя плоских заготовок, система ЧПУ в качестве входной информации может использовать текстовый файл в формате обмена данными — например, DXF или HPGL.

DXF (англ. ***D****rawing* ***eX****change* ***F****ormat*) — открытый формат файлов для обмена графической информацией между приложениями САПР. Был создан фирмой Autodesk для системы AutoCAD. Поддерживается практически всеми CAD-системами на платформе PC.

HPGL (иногда *HP-GL*) является основным языком управления принтерами, используемым плоттерами Hewlett-Packard. Его название представляет собой аббревиатуру ***H****ewlett-****P****ackard* ***G****raphics* ***L****anguage*. В данный момент он является стандартным почти для всех плоттеров.

На сегодняшний день на рынке существуют пульты, которые способны только задавать настройки режимов обработки и перемешать рабочими инструмент в ручном режиме, но нет пульта, который способен считывать траектории с внешних носителей информации и преобразовывать эту информацию в машинный код для выполнения непосредственной самой программы изготовления конечной продукции. Так как для этих задач используют промышленный компьютер, на котором можно в случае необходимости подкорректировать управляющую программу. Но с развитием технологий управляющие программы уже на стадии написания можно испытать на виртуальных станках и обнаружить все недочеты и отклонения в работе, так что данный функционал корректировка в процессе производства уже не нужна. От ЧПУ требуется только считывание и конвертирование программ управления для управления станком.

1.3 Выбор микроконтроллера

На большинства, производствах для работы ЧПУ станков используется в качестве системы управления промышленный компьютер (ПК), который выполняет роль считывания параметров обработки и траекторий движения рабочего механизма из написанной на специальном языке программирования программы и передаче их станку. Однако, данный ПК достаточно дорогостоящие и занимаю приличное место, если выполняемая программа не подразумевает частое изменение и редактирование параметров и траекторий то можно использовать достаточно дешевый и компактный пульт управления на микроконтроллере, который будет считывать из памяти программу и преобразовывать её в машинные команды для изготовления конечной продукции. Для выбора нужного микроконтроллера который отвечал бы всем требования ознакомимся с причиной появления, историей создания и молельным рядом который представлен сейчас на рынке[20].

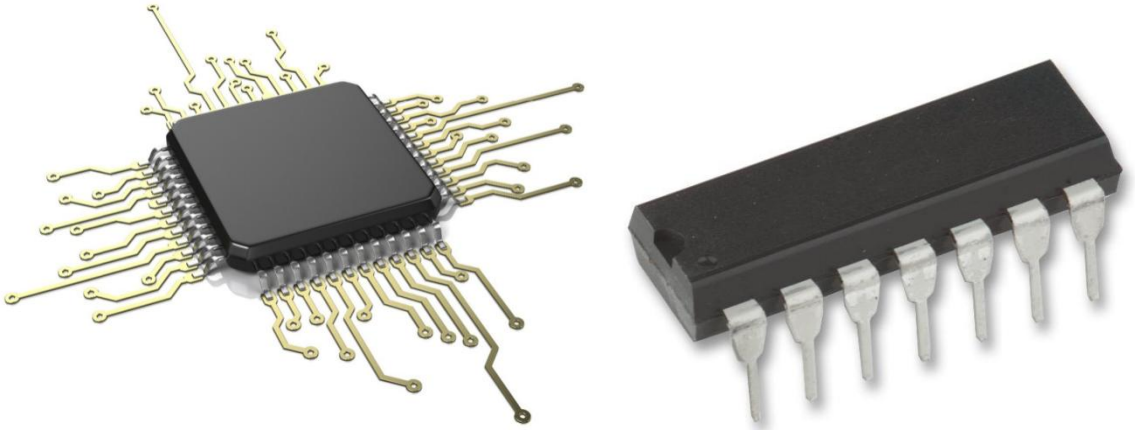

Рис. 3. Микроконтроллеры

Одна из основная причин стимулирующая появление микроконтроллеров - внедрение средств вычислительной техники во все сферы человеческой деятельности, в том числе и в промышленной. Это повлекло миниатюризации и снижения стоимости изделий. Прогресс же в

технологии создания микросхем в значительной степени способствовал этому.

В 1967 г. фирма Texas Instruments выпустила 1 калькулятор на интегральной микросхеме, который и запустил процесс миниатюризации средств вычислительной техники. Однако микросхемы для калькуляторов создавались применительно к требованиям каждого конкретного заказчика. Эти процессоры для калькуляторов были маломощными и не отвечали требованиям, предъявляемым к вычислительным мощностям[21].

Работая над заказом одной из японских фирм над разработкой программируемых калькуляторов, специалист фирм Intel предложил новую концепцию проектирования микропроцессоров, которая заключалась в создании микропроцессоров общего назначения, способных выполнять абсолютно любые арифметико-логические операции. В 1969 г. фирмой Intel был представлен первая в мире специализированный интегральный чип Intel 4004 промышленный серийный выпуск с 1971 г., который по сути, являлся 4-разрядным программируемым микропроцессором. Микропроцессор Intel 4004 содержал примерно 2250 транзисторов и мог выполнять примерно 60 тыс. операций в секунду. Производительность Intel 4004 была слишком низкой, чтобы он мог в полной объеме выполнять функции процессора. Однако он мог служить основным микропроцессором для калькуляторов. Интересно факт, что название фирмы Intel происходит от слов "Integrated Electronics" (интегральная электроника). Возможно, смысловое содержание названия Intel играет не малую роль в том, что фирма с момента своего создания (1968 г.) и по нынешнее время (сегодня в фирме Intel работают около 65 тыс. сотрудников) занимает лидирующее положение среди мировых производителей микросхем.

В 1976 г. создан первый 8-разрядный микроконтроллер, на кристалле которого были интегрированы основные элементы микропроцессорной управляющей системы: процессор, память типа ROM и RAM, порты ввода/вывода и таймеры.

В 1978г. появился 16-разрядный микропроцессор Intel 8086 (отечественный аналог К1810ВМ86) - первый микропроцессор массового использования. С серийного выпуска Intel 8086 началось масштабное проникновение персональных компьютеров во все сферы деятельности. Созданию микропроцессора Intel 8086 предшествовал выпуск 8-разрядных микропроцессоров Intel 8008 (1972 г.) и Intel 8080 (1974 г.)[22].

Создание фирмой Texas Instruments в 1982 г. первого однокристального программируемого цифрового сигнального процессора (DSP) TMS32010. В текшее время из-за сложности алгоритмов и специфики математических операций, используемых при создании интеллектуальных приводов электрических двигателей, архитектура сигнальных процессоров используется в качестве основных для современных универсальных программируемых DSP-контроллеров, широко применяемых в системах управления электродвигателями всех типов.

Структурная схема любой электронно-вычислительной машины содержит следующие блоки: процессор, состоящий из арифметико-логического устройства (ALU), схем управления и регистров; память; периферийные устройства ввода/вывода данных.

Первоначально блоки, входящие в ЭВМ, создавались на базе стандартных дискретных логических микросхем, которые выполняли сравнительно простые функции.

Поэтому сами электронно-вычислительной машины имели большие габариты (например, суперЭВМ Сгау-1 состояла из 300 тыс. микросхем и занимала объем порядка трех кубических метров).

Успехи интегральной технологии привели к появлению больших и сверхбольших интегральных схем (БИС и СБИС) с размещением до 10 и 100 тысяч, а в настоящее время - 10 миллионов транзисторов на одном кристалле. Высокая степень внедрения БИС и СБИС позволила в одной микросхеме реализовать отдельные блоки ЭВМ, к примеру, процессор.

В микросхемах первых микропроцессоров (например, Intel 8080) были реализованы только сам процессор и дополнительные устройства, осуществляющие управление обменом данными с внешней памятью и устройствами ввода/вывода данных.

Однако, кроме устройств, входящих в состав процессора, на кристалле СБИС (БИС) могут быть выполнены память для хранения программ (ROM), данных или промежуточных результатов (RAM), периферийные устройства ввода/вывода. Такие СБИС (БИС) относятся к классу однокристальных микро ЭВМ. Одной из 1 серийно выпускаемых микросхем однокристальных микро ЭВМ стала Intel 8048 (18048). Однокристальные микро ЭВМ начали активно использоваться там, где была потребность в несложной цифровой обработке данных: бытовых приборах, простых системах управления, контроля и т. п. Кроме однокристальных микро ЭВМ для цифровой обработки в подобных системах можно использовать и заказные, специализированные интегральные микросхемы (Application Specific Integrated Circuit - ASIC).

Идея интеграции на одном кристалле совместно с процессором и памятью большого количества стандартных устройств, всевозможных назначения воплотилась в появление микроконтроллеров. Одним из 1 серийно выпускаемых микроконтроллеров можно считать Intel 8051. Вскоре микроконтроллер 8051 завоевал популярность. В настоящее время микроконтроллеры с набором команд 8051 выпускаются 10 фирм-производителей Analog Devices, Atmel, Dallas, Semiconductor, Oki, Philips, Infineon Technologies, Silicon Storage Technologies, Temic и многими другими.

В отличие от универсальных микропроцессоров, предназначенных в основном для математической обработки данных, микроконтроллеры имеют расширенный набор встроенных периферийных устройств. Это могут быть доп. блоки памяти типа RAM, ROM, EPROM, EEPROM или FLASH. Периферийные устройства всевозможного назначения: универсальные таймеры и таймеры специального назначения; "сторожевые" таймеры;

контроллеры внешних интерфейсов (UART, USART, SPI, SCI, PC, j 1850, USB- или CAN-шины) и жидкокристаллических дисплеев; монитор источников питания; аналоговые и цифровые компаратора; схему перезапуска; аналого-цифровые (АЦП) и цифро-аналоговые (ЦАП) преобразователи и др. Таким образом, микроконтроллеры содержат все периферийные устройства, нужные для создания законченных встроенных систем управления, контроля и что немаловажно для дальнейшего понимания, стандартные устройства, которые в случае использования микропроцессора в системе выполнялись бы на базе дополнительных внешних по отношению к микропроцессору микросхем. В микроконтроллерах по крайней мере, большая часть периферийных устройств выполнены на одном кристалле с процессором, что придает системам на базе микроконтроллеров значительно большую гибкость и универсальность. Примером подобных высокоинтегрированных микроконтроллеров могут служить микроконвертеры ADcU812/814/816/824/834 (Analog Devices), H8/300L (Hitachi), 51XA-G49 и 51XA-G3 (Philips), MC68HC05/08/ 11/12/16, микроконтроллеры семейств AT89/AT 90 (AVR)/AT tiny/AT mega (Atmel), C16x (Infineon), H8/300 и (Motorola), PICmicro (Microchip), MSP430F (Texas instruments) и многие другие[23].

Что такое микроконтроллер

Прежде всего, разберемся с самим понятием «микроконтроллер». Микроконтроллер можно определить как миниатюрный компьютер на базе одного-единственного чипа, включающий, помимо процессора ряд вспомогательных элементов, таких, как ОЗУ, ППЗУ, таймер, и.т.д. Микроконтроллер предназначен для выполнения каких-либо заранее определенных заданий.

Проще всего сравнивать микроконтроллер с ПК. Как и ПК, микроконтроллер имеет процессор, ОЗУ и постоянную память. Однако, в

отличие от персонального компьютера, все эти элементы расположены на одном чипе.

ПК создан для того, чтобы выполнять грамозкие задачи общего назначения. Например, вы можете использовать компьютер, для набора текста, написания программ, хранения и запуска видео файлов, путешествия в Интернет, и.т.д. Микроконтроллеры предназначены для выполнения конкретных заданий, например, коммутации кондиционера, когда температура в помещении опускается ниже определенного значения, или наоборот выше.

Существует несколько популярных семейств микроконтроллеров, которые используются для всевозможно различных целей. Наиболее распространенными из них являются семейства микроконтроллеров 8051, PIC и AVR. И последние в настоящее время набирают особую популярность на счет проекта Arduino «Микроконтроллеры это просто». В следствии чего документации на них достаточно много и существует большое сообщество по поддержке начинающих разработчиков и как следствие популяризации цены на них значительно меньше чем у конкурентов.

История семейства

Семейство микроконтроллеров AVR было основано в 1996 г. компание Atmel, а разработчиками архитектуры являются Alf-Egil Bogen и Vegard Wollan. Отсюда и происходит название семейства – от первых букв имен разработчиков – A и V, и первой буквы аббревиатуры RISC – типа архитектуры, на которой основана архитектура микроконтроллеров. Также это сокращение часто расшифровывают как Advanced Virtual RISC (модернизированный эффективный RISC).

Одним из первым микроконтроллером в серии был AT90S8515, однако 1 микроконтроллером, выпущенным на рынок, стал AT90S1200. Это произошло в 1997 г.

На сегодняшний момент, на рынке доступны 3 линейки микроконтроллеров:

- TinyAVR – малый объем памяти, малые размеры, подходит для самых элементарных задач.

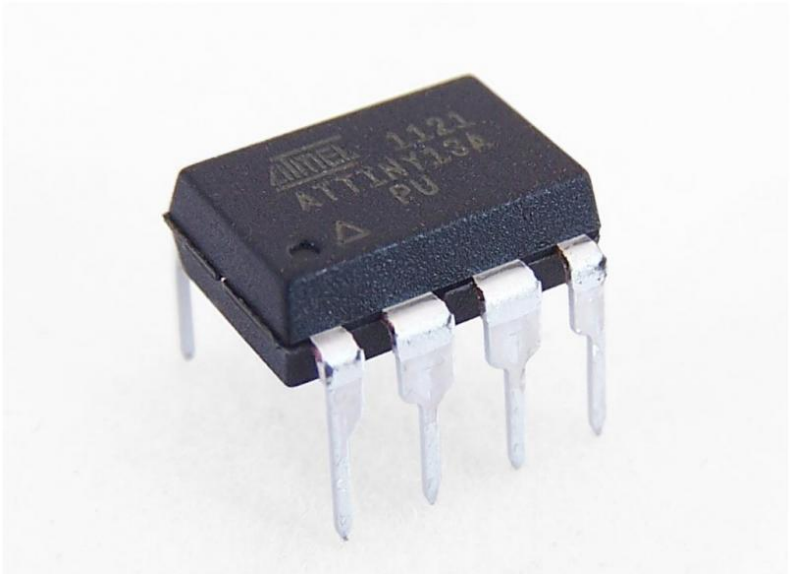

Рис.4. Внешний вид микроконтроллера TinyAVR

- MegaAVR – наиболее популярная линейка, имеющая достаточный объем встроенной памяти (до 256 КБ), множество дополнительной переферии и служит для задач средней и высокой сложности.

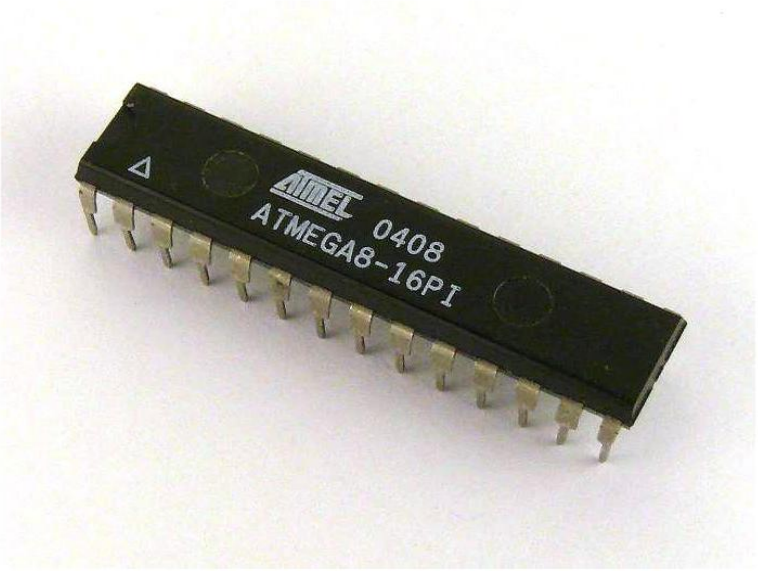

Рис. 5. Внешний вид микроконтроллера MegaAVR

- XmegaAVR – используется в продвинутых коммерческих задачах, требующих большого объема памяти и высокой производительности.

Рис.6. Пример микроконтроллера XmegaAVR

Таблица 1. Сравнительные характеристики различных линеек:

Название серии	Число контактов	Объем флэш-памяти	Особенность
TinyAVR	6-32	0,5 – 8 КБ	Небольшой размер
MegaAVR	28-100	4-256 КБ	Периферийные устройства
XmegaAVR	44-100	16-384 КБ	Система прерываний, поддержка DMA

1.4 Особенности семейства

Прежде всего, микроконтроллеры этой серии являются высокоскоростными. Многие инструкций процессор микроконтроллера выполняет за один цикл. Микроконтроллеры AVR примерно в 4 раза быстрее, чем PIC. Кроме того, они потребляют значительно меньше энергии и могут работать в нескольких режимах экономии энергии.

Многие контроллеров AVR являются 8-разрядными, хотя есть и 32-разрядная разновидность контроллеров AVR32. Кроме того, как уже говорилось выше, AVR принадлежат к типу RISC-микроконтроллеров. Архитектура RISC (Complex Instruction Set Computers) означает, что набор инструкций, которые может выполнять процессор устройства, является урезанными, но, в то же время, данная архитектура дает преимущество в быстродействии. Аналогом архитектуры RISC является архитектура CISC (Complex Instruction Set Computers).

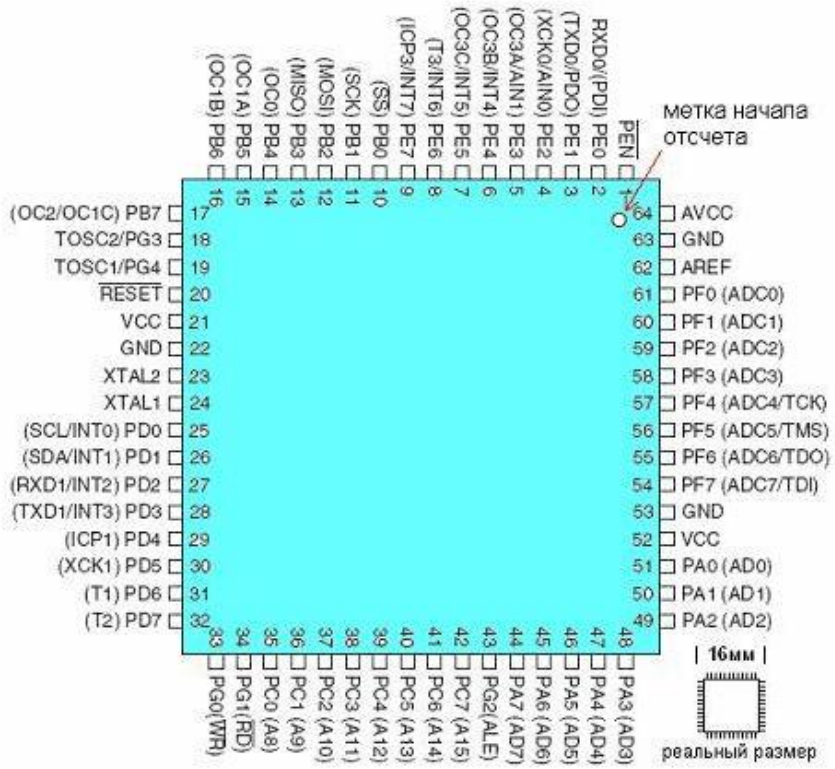

Рис.7. 32-разрядная разновидность контроллеров AVR32

8-битность контроллера подразумевает, что он способен передавать и принимать 8-битные данные. Предоставленные регистры ввода/вывода также являются 8-битными.

Архитектура контроллера базируется на регистрах. Это означает, что для хранения исходных значений операции и результата в контроллере применяются регистры.

Процессор контроллера получает данные из двух входных регистров, выполняет логическую операцию и сохраняет результат в выходном регистре. Все это занимает 1 исполняемый цикл.

Архитектура контроллера

Всего контроллер AVR имеет 32 8-битных регистра общего применения. В течение цикла процессор берет данные из двух регистров и использует их в арифметико-логическое устройство (АЛУ), которое производит вычисление над данными и помещает их в случайный регистр. АЛУ может выполнять как арифметические, так и логические действия над регистрами. Также АЛУ может выполнять и действия с одним регистром. При этом контроллер не имеет операнда-аккумулятора, в отличие от контроллеров семейства 8051 – для операций могут использоваться любые операнды, и результат операции также может быть помещен в любой операнд.

Микроконтроллер соответствует Гарвардской вычислительной архитектуре, согласно которой компьютер имеет независимую память для программ и данных. Поэтому в то время, пока выполняется одна операция, происходит предварительное извлечение из памяти следующей операции.

Котроллер способен выполнять одну операцию за цикл. Из этого следует, что если тактовая частота микроконтроллера составляет 1 МГц, то его производительность составит 1 млн. операция в секунду. Чем выше

тактовая частота контроллера, тем выше будет его производительность. Однако при выборе тактовой частоты контроллера следует соблюдать компромисс между его скоростью и энергопотреблением.

Помимо флэш-памяти и процессора контроллер имеет такие периферии, как порты ввода-вывода, аналого-цифровой преобразователь, таймеры, коммуникационные интерфейсы – I2C, SPI и последовательный порт UART. Все эти периферии могут контролироваться на программном уровне.

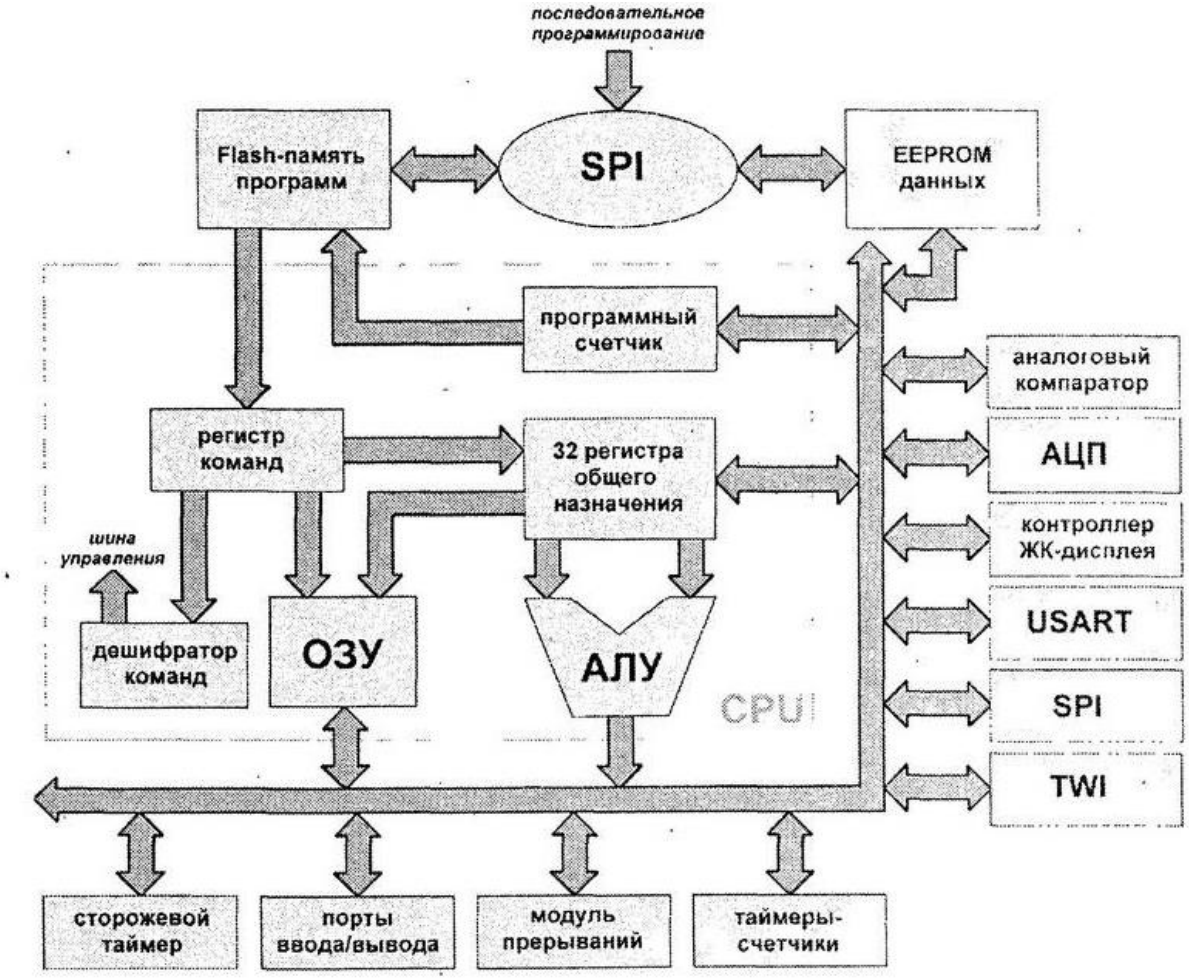

Рис.8. Типовая архитектура микроконтроллеров AVR

Программы для микроконтроллера

Как уже упоминалось выше, микроконтроллер подобен ПК, а из этого следует, что, как и ПК, AVR также может выполнять какую-либо программу, хотя и всего одну в определенный момент времени.

Программа микроконтроллера может храниться во внутренней памяти контроллера и представляет собой серию элементарных команд, которые выбирают данные и осуществляют с ними некие операции. В некоторых случаев это означает считывание входящих данных, проверка их состояния и вывода соответствующих выходных значения. Иногда может потребоваться изменение данных и совершение с ними операций, а также передача данных какому-либо внешнему переферийному устройству, например, индикатору, или последовательному порту.

Для таких простых задач используются наборы элементарных команд, каждая из которых имеет аналог на более доступном человеческому восприятию языке. Поэтому наиболее распространенным способом написания программ для контроллера является написание их на языке машинных команд.

Преимуществом машинных команд является очень быстрый, компактный и эффективный код, но создание таких программ одновременно требует и обширных знаний работы процессора микроконтроллера, ручного управления памятью и контроля структуры программы. Поэтому зачастую для написания программ используются другие языки высокого уровня, такие, как Basic, C, и Java. В этом случае задачу по контролю структуры программы и управлению памятью берет на себя компилятор который создает прошивки. Кроме этого, часто используемые функции могут быть при этом помещены в специальные библиотеки и извлекаться из них по мере необходимости[24].

Микроконтроллеры семейства AVR на сегодняшний день широко используются в компьютерах, для автоматизации управления электронной техникой, различными электроприборами и механизмами, применяемыми в промышленных, коммерческих, а также бытовых нуждах. Невысокая стоимость, большой ассортимент и широкие возможности микроконтроллеров этой серии способствовали их большой популярности.

1.5 Выбор платформы микроконтроллера.

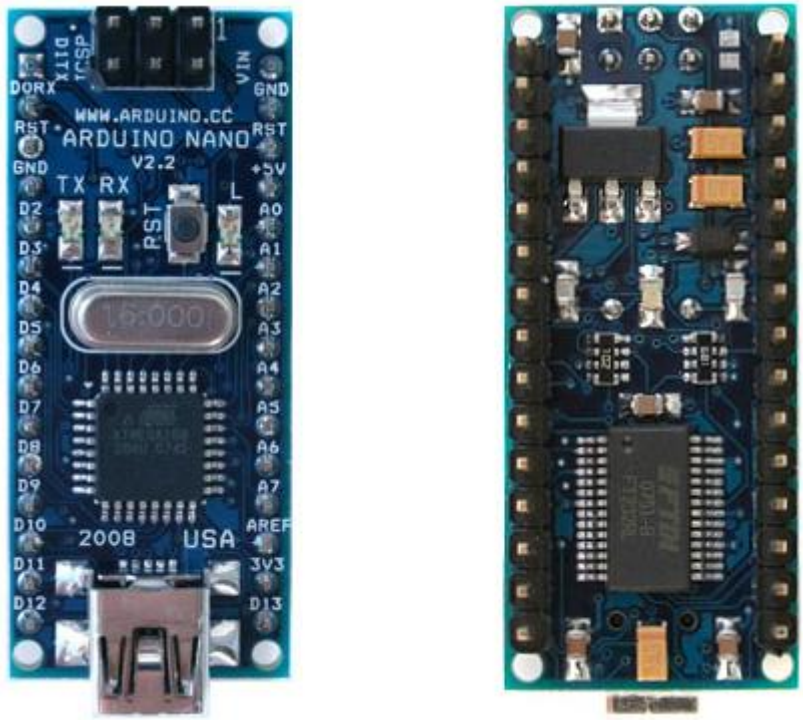

Рис.9. Ардуино Нано

Общие сведения

Платформа Nano, построенная на основе микроконтроллере ATmega328 имеет небольшие размеры и может применяться в различных проектах. Она имеет схожую с Arduino Duemilanove структуру, однако отличается внешним видом. Отличие заключается в нехватке основного разъема постоянного напряжения питания и работе через кабель MiniUSB.

Основные параметры платформы

Микроконтроллер	Atmel ATmega328
Рабочее напряжение	5 В
Входное напряжение	7-12 В
Входное напряжение	6-20 В
Цифровые Входы/Выходы	14
Аналоговые входы	8
Постоянный ток через выводы	30 мА
Постоянная память	32 Кб (2 Кб для загрузчика)
ОЗУ	2 Кб

EEPROM	1 Кб
Стандартная частота	16 МГц
Размеры	1.9 см x 4.3 см

Питание:

Питается Arduino Nano через информационный кабель MiniUSB, или от нестабилизированного источника питания 6-20 В (вывод 30), или от стабилизированного 5 В (вывод 27), внешнего источника постоянного напряжения. Автоматически используется источник питания с наивысшим напряжением питания.

Микросхема FTDI FT232RL получает питающее напряжение, только если платформа получает напряжение питания от USB. Что позволяет работать от внешнего источника, напряжение 3.3 В будет отсутствовать, так как оно формируется микросхемой FTDI, при этом светодиоды RX и TX загораются только при появлении сигналов на выводах 0 и 1.

Входы и Выходы

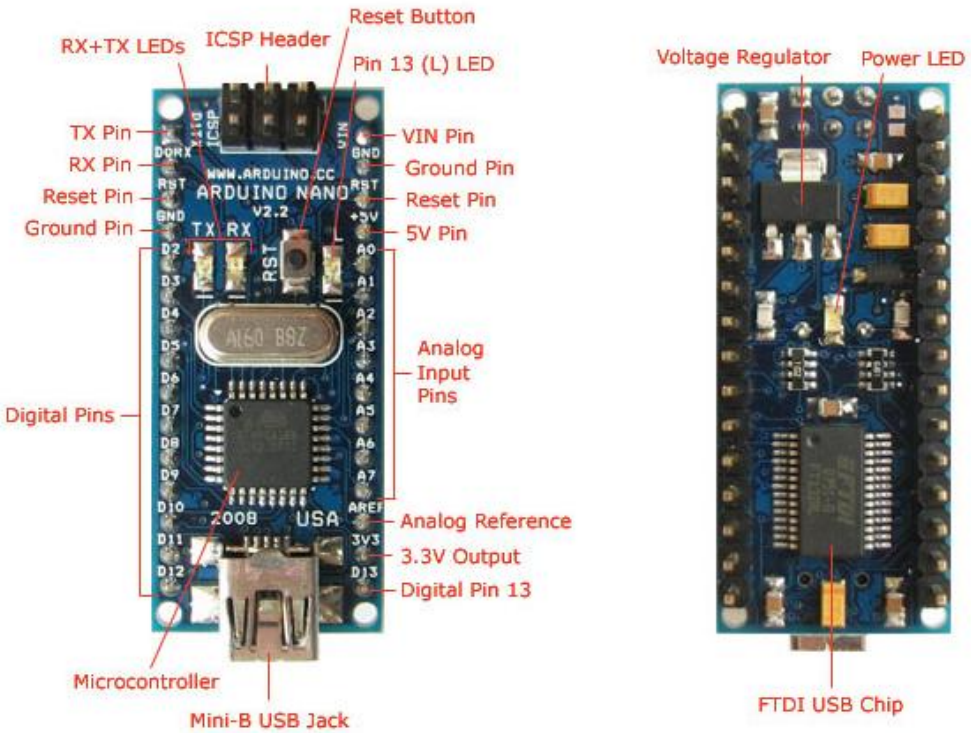

Рис.10. Выводы микроконтроллера

Каждый из 14 цифровых выходов, используя функции pinMode(), digitalWrite(), и digitalRead(), может компилироваться в программе, как вход, так и выход. Выводы работают только при наличии питающего напряжения 5В. Каждый вывод имеет подтягивающий резистор сопротивлением примерно 10-40 кОм и может выдавать на выходе до 30 мА. Некоторые из имеющих выводов имеют специфичные функции:

- **Последовательный Serial порт: 0 (RX) и 1 (TX).** Используются для обмена, получения и передачи данных.

- **Внешние прерыватели: 2 и 3.** Данные выводы также могут быть использованы на прерывания при низком значении или на предельном, заднем фронте, или при изменении значения.

- **ШИМ: 3, 5, 6, 9, 10, и 11.** Каждый из выводов задает ШИМ со значением до 8 бит за счет специальной функции analogWrite().

- **SPI: 10 (SS), 12 (MISO), 11 (MOSI), 13 (SCK)**. При помощи данных контактов реализована связь SPI.

На данной платформе размещены восемь аналоговых контактов, каждый имеет разрешение 10 бит. Стандартно выводы обладают диапазон измерения до 5 В относительно нуля, не смотря на это имеется способность изменить верхнее значение посредством специализированного функционала analogReference(). Так же имеются контакты со специфичными функциями:

- **I2C: A4 (SDA) и A5 (SCL).** Посредством команд выводов могут осуществляется связь по интерфейсу I2C (TWI).

Также на плате разведены пара выводов:

- **AREF.** Необходимо для отдельно питания аналоговых входов. Используется с командой analogReference().

- **Reset**. Для перезагрузки микроконтроллерного процессора. Обычно, как правило, применяется для присоединения кнопки перезапуска на платах расширения, перегораживающий доступ к кнопке непосредственно на самой плате.

Связь

На данной платформе установлено множество устройств, для реализации связи с компьютером, а также другими устройствами Ардуино или микроконтроллерами иного семейства. ATmega328 обладают последовательным интерфейсом UART TTL (5 В), реализующий контактами RX и TX. Расположенная на плате данная микросхема направляет данный интерфейс через USB, а драйверы предоставляют имитирующий COM порт программе на персональном компьютере. Монитор последовательной шины (Serial Monitor) программы Ардуино позволяет отправлять и принимать текстовые значения при подключении. Светодиоды RX и TX на платформе будут указывать обмен информации по TX и RX.

ATmega328 также поддерживают распростанненые интерфейсы I2C (TWI) и SPI. В Ардуино имеется библиотека Wire предоставляющая удобства применения шины I2C.

Программирование

Платформа может программироваться на языке высокого уровня C++, но также для удобства начинающих программируется посредством специального упрощенного ПО Arduino 1.6.5. Для этого в меню Инструменты > Board выбираем «Arduino Nano, Микроконтроллер ATmega328 поставляются со всем необходимым, упрощающий запись программ без использования внешних программаторов. Связь реализована по протоколу STK500.

Имеется также возможность не прибегать к загрузчику и запрограммировать платформу через внутрисхемному протоколу выводами блока ICSP [25].

Автоматическая (программная) перезагрузка

Nano спроектирован таким образом, что перед записью последующего программного кода перезагрузка реализована самой программой, а не нажатием специальной кнопки на плате. Одна из ножек FT232RL, осуществляемая управлением протоколом данных (DTR), подключена к выводу перезагрузки ATmega328 через защитный конденсатор 100 нФ. Запуск данной линии, перезагружает процессор микроконтроллера. Программа Arduino, задействуя данный функционал, загружает код нажатием кнопки Загрузить в самой среде разработки. Подача сигнала низкого значения по линиям DTR синхронизирована с началом текущей записи программы, что сокращает время загрузчика.

Функция применяет еще одно назначение. Перезагрузка Ардуино происходит постоянно при подключении к программе Ардуино1.6.5 на персональном компьютере. Следующие несколько секунд после перезагрузки запускается загрузчик. Во время прошивки происходит таймаут нескольких байтов исходного кода во избежание получения ошибочных данных. Если производится однократная отладка прошивки, записанной в плату, или ввод каких-нибудь других данных при каждом запуске, необходимо удостоверится, что программа на компьютере ждет в течение первой секунды перед передачей данных[26].

Выводы платформы микроконтроллера

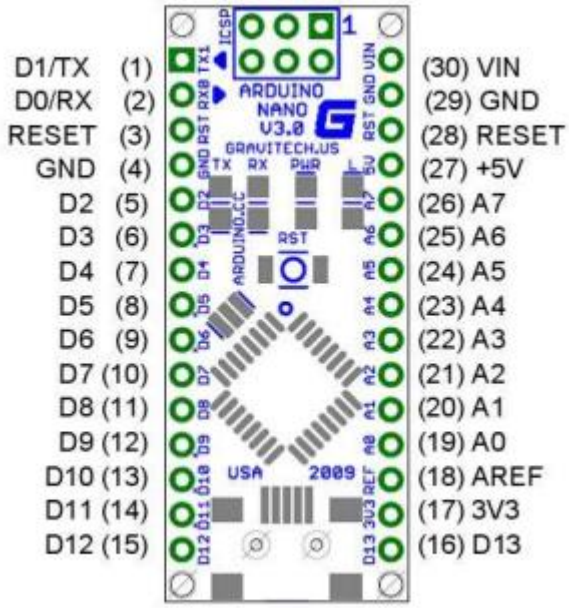

Рис.11. Обозначение выводов МК

Таблица 2. Выводы ARDUINO NANO

№ вывода	Наименование	Тип	Описание
1 - 2, 5 - 16	D0 - D13	Ввод - вывод	Цифровой ввод-вывод портов D0 - D13
3, 28	RESET	Ввод	Сброс (активный уровень - низкий)
4, 29	GND	Питание	Общий питания
17	3V3	Вывод	+3,3 В с микросхемы FT232
18	AREF	Ввод	Опорное напряжение АЦП
19 - 26	A0 - A7	Ввод	Аналоговый вход, каналы 0 - 7
27	+5V	Ввод - вывод	+5 В на вывод от регулятора на плате, или +5 В на вход от внешнего источника питания

1.6 Программа управления чпу станком

Разрабатываемый автономный чпу станок будет работать под управлением специального программного обеспечения, которое должно уметь считывать параметры и траектории обработки разработанной заготовки. Для этого необходимо выбрать на каком языке будут писаться все необходимые параметры.

На сегодняшний день существует более 100 языков для написания программ станкам с ЧПУ, но до сих пор не существует единого языка, который в достаточной мере удовлетворял бы всем необходимым требованиям. Языки отличаются степенью автоматизации и степенью специализации, самый распространенный язык G-code

G-код — условное наименование языка для программирования устройств с числовым программным управлением. Этот язык был создан компанией Electronic I.A. в конце 1950-х , начале 1960-х. Окончательная доработка и одобрение произошло в феврале 1980 года и было принято стандартом RS274D. Комитет стандартов ISO принял G-код стандартом ISO 6983-1:2009. Государственный комитет по стандартам СССР принял его как ГОСТ 20999-83. В советской нормативной документации G-код именуется кодом ИСО 7-бит (ISO 7 bit). Создан для передачи информации ЧПУ в виде кода написанного машинным языком, аналогично как и коды PC8C или AEG.

Изготовители систем с числовым программным управлением, применяют программное обеспечение управления станком, для которого написана программа обработки деталей в качестве основной команды управления, G-код используется в виде основного языка программирования, увеличивая его на своё усмотрение. Ниже представлено подробное описание G-кода с примерами

G00 –перемещение инструмента при максимальной скорости. Код G00 применяется для перемещения рабочего механизма от одного

обрабатываемого участка к другому. Применяется для максимально быстрого перемещения инструмента для резки к искомому месту обработки детали или к позиции безопасности. Для выполнения обработки детали никогда не применяется ускоренное перемещение инструмента, потому что скорость движения исполнительного механизма станка очень довольно быстрая и меняется на протяжении всей работы . Код G00 сменяется другой командой при ипримененении следующих кодов: G01, G02, G03.

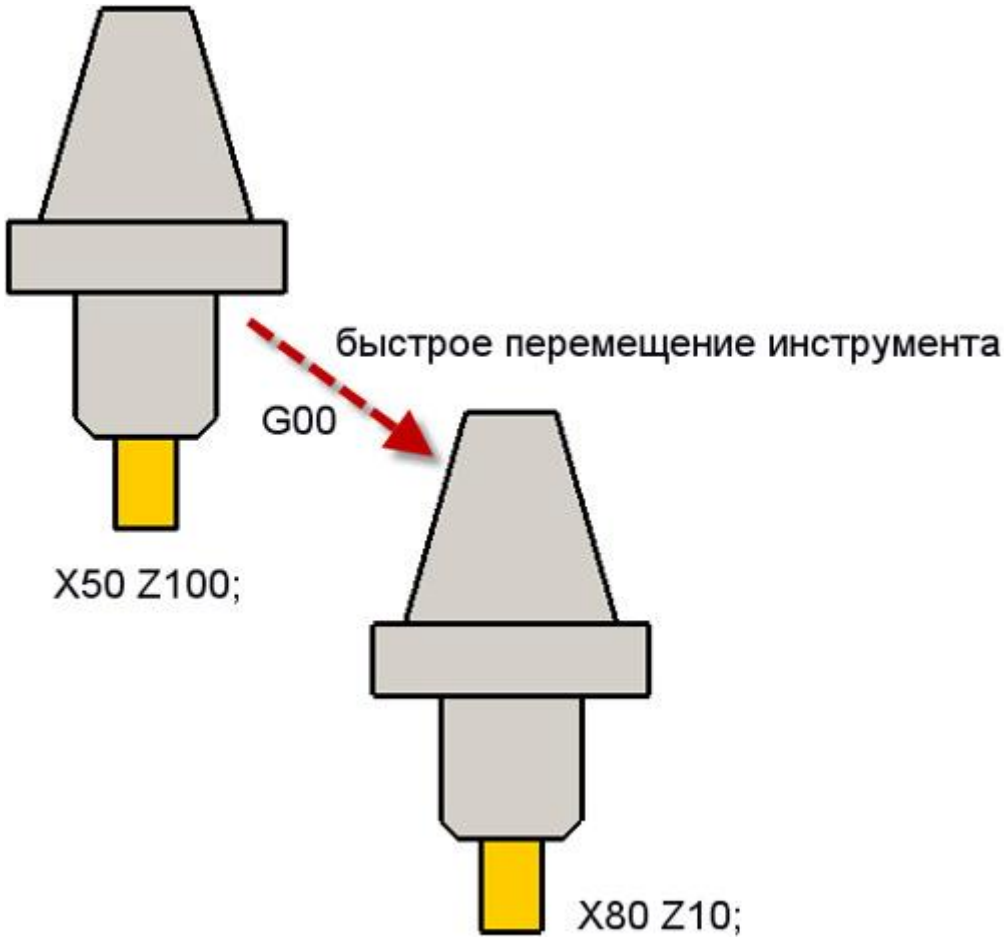

Рис.12 Пример ускоренного перемещения инструмента. G00 X80 Z10 – перемещение в точку с координатами (10; 80).

G01 – линейная интерполяция. Код G01 – исполнительная команда позволяющая осуществить перемещение исполнительного механизма по прямой с заранее установленной скоростью. Скорость перемещения инструмента задается адресом - F. Код G01 отменяется с помощью кодов G00, G02HG03.

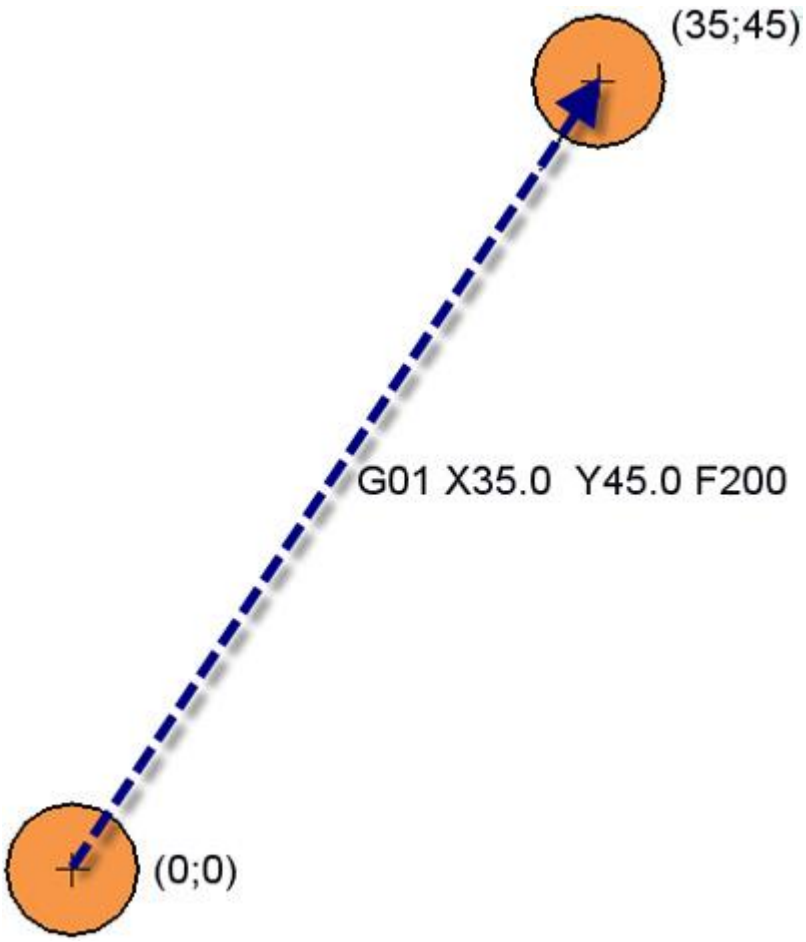

Рис. 13 Пример обеспечения линейной интерполяции инструмента. G01 X35 Y45 F200 – перемещение инструмента в точку с координатами (35; 45) по прямой со заданной скоростью подачи 200 миллиметров в минуту.

G02 – круговая интерполяция (дуга в направлении по час. стр.). Код G02 используется для исполнения интерполяции с кругом, иными словами для передвижения инструмента по кругу в направлении часовой стрелки с заранее заданной скоростью хода. Скорость перемещения заранее принимается адресом - F. Код G02 отменяется кодами G03, G00 , G01.

G03 – перемещение по кругу (против час. стр.). Код G03 используется для перемещения исполнительного инструмента по кругу, иными словами для позиционирования исполнительного инструмента по кругу против часовой стрелки с заранее установленной скоростью. Скорость перемещения задается адресом - F. Код G03 снимается кодами G02, G01 ,G01.

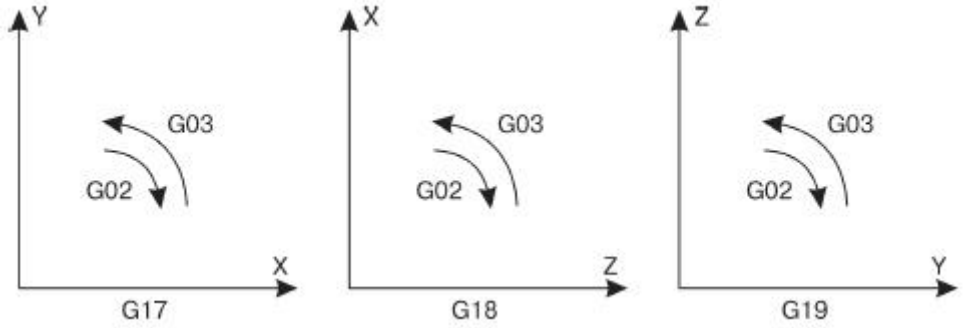

Рис. 14. Интерполяция в круге на различных плоскостях

G04 – пауза. Исполняемый параметр задает выдержку с интервалом времени задаваемым заранее выполняется кодом G 04 . Этот программный код выполняется вместе с P - а так же адресом - X , он задает временя паузы. Время находится в интервале между 0.001 и 99999.999 секунадми. Код G04, P - или адрес - X выполняются в одном заходе , он не должен иметь каких либо перемещений по оси координат.

Когда необходимо определить временя задержки, используется команда P, в этом случае нельзя программно использовать десятичную точку. Адрес с индексом P назначает выдержку по времени в миллисекундах, а индекс X назначает выдержку в секундах. Когда командный код G04 устанавливается без времени задержки импульса, она принимается станком как команда для точной остановки инструмента.

Пример:

G04 X1.6 – задержка 1.6 секунды;
G04 P2500 – задержка 2.5 секунды.

G09 – остановка точная. Из-за увеличения скорости и соответственно её замедления относительно осевых переходов исполнительных механизмов станка , отсутствует точная выполнение среза угловых кромок при перемещении команды резания от одной к другой. Этот недочет обработки изделия выглядит как в притуплении и закруглении углов.

Допустим, когда вы обрабатываете по периметру контур в виде прямоугольника и пытаетесь сделать кромку в углу очень острую (рис. 6). Работая в обычном режиме , наверняка , что при перемещении от движения по оси X к перемещению по оси Y выйдет скругление кромок (рис. 7). Очень сильно этот вредный эффект виден когда станок работает при очень высоких скоростях подачи и при работе в огромных центрах обработки.

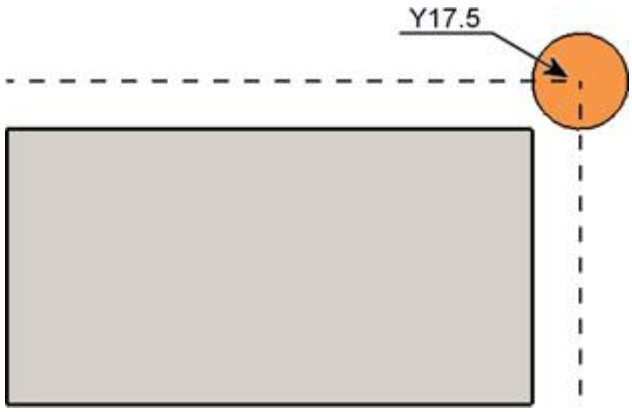

Рис. 6. Получение острой кромки в правом углу

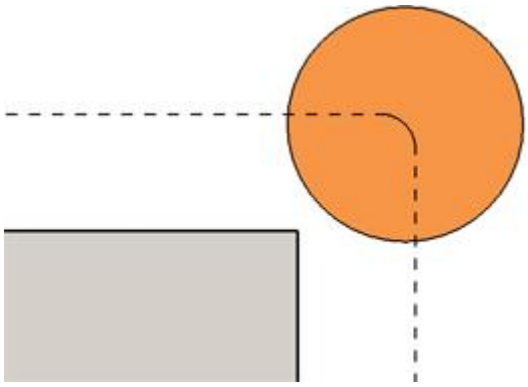

Рис. 7. Обрезка методом скругления кромки

G09 это код немодальный, он используется когда необходимо согласование текущей траектории инструмента станка с заданной заранее траекторией. При перемещении из одного движения к другому станок совершит конечное перемещение в заданную позицию.

G09 указывается вместе с заданной координатой, для которой будет выполнено точное позиционирование. Управляющая программа,

гарантирующая получение острой кромки угла прямоугольного контура, будет выглядеть так:

```
%
O0005
N100 G21
N102 G0 G17 G40 G49 G80 G90
N104 T1 M6
N106 G0 G90 G54 X30. Y-22.6 S1000 M3
N108 G43 H1 Z100.
N110 Z10.
N112 G1 Z-2. F90.
N114 Y-12
N116 G09 Y17.7
N118 X-25.
N120 X-35.
N122 Z8.
N124 G0 Z90.
N126 M5
N132 M30
%
```

Когда инструмент приходит в координату Y17.7, то станок выполняет точную остановку. Время задержки в этой позиции определяется значением параметра системы.

G10 – ввод данных управления станок. Команда G10 устанавливает и изменяет систему координат в работе и вводит заданные параметры для корректировки инструмента станка с помощью программы управления.

При желании внести иные параметры для корректировки с помощью программы управления, их необходимо разместить в начале кода. Таким образом, происходит согласование параметров корректировки и изменения в программе управления.

Как правило для ввода параметров корректировки используется следующий вид кода:

G10 L11 P_ R _ ;

где G10 – режим ввода параметров ; L11 – настройка корректировки инструмента станка ; P – выбор корректировки , который надо изменить; R – вносимое значение.

Если команда G10 выполняется вместе с командой G90, то параметры в регистрах корректировки будут изменены. Если G10 работает совместно с командой G91, то параметры в корректорах складываются или вычитаются с значением R. Например , G10 G90 Lll P13 R91.14 меняет действительное значение в корректировке № 13 на новое значение 91.14.

При необходимости установить или сместить рабочую координатную систему, выполняется следующая команда:

G10 L2 P _ X _ Y _ Z _ ;

где G10 – режим ввода данных , его включение; L2 – установка рабочей стандартной системы координат; P – инициализация системы координат; X, Y, Z – параметры, выбирающие заданное расположение рабочей системы координат.

Предварительное значение G10 будет модальной и будет ею до того времени, пока не поступит команда отмены G11. Прежде чем использовать G10 необходимо внимательно изучить документацию к устройству, потому что параметр G10 бывает разнообразным по значению.

G11 – означает, окончание режима ввода данных в станок. С помощью данной команды G11 прекращает свое действие G10 для начала режима ввода параметров в станок.

G15 – отменяет режим ввода данных полярных координат. С помощью команды G15 вы деактивируете установку в полярной системе координат и переходите назад к вводу данных с помощью систему координат в прямоугольной форме.

G16 – включает режим полярных координат. Данная команда G16 будет исполняться в полярной системе координат. В связи с чем применяется позиционирование определяемое зависимостью расстояния и угла от точки нуля данной координатной системы или от текущей позиции инструмента.

Использование станка в полярной системе координат разрешается в каждой из трех осей системы. Кодом программы G17 вы осуществляете перемещение в осях XY, с кодом программы G18 – в осях XZ, а при помощи кода программы G19 – в осях YZ.

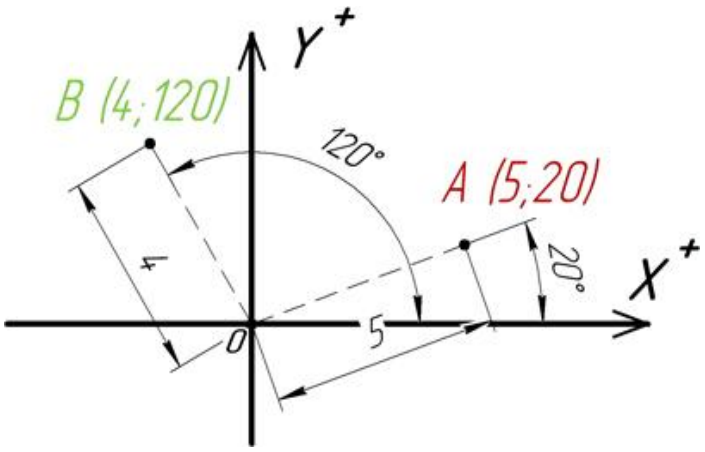

Рис. 15 Полярные координаты: точка A (5;20) и точка B (4; 120)

Когда выбирается данная плоскость осей XY, то X - адрес указывает на радиус, а Y выбирает относительно оси X его угол. Когда используется ось XZ, то адрес X обозначает радиус, а адрес Z обозначает относительный угол оси X. Когда заданная ось YZ, то адресное поле - Y задает радиус, а Z задает относительный угол оси Y. Угол отсчитываемый против часовой стрелки , является положительным.

Перемещения в полярной системе координат, на которые указывают действующая команда G90, исполняются строго относительно нулевой точки настоящей системы координат в работе. При исполнении кода программы G91, перемещения в полярной системе координат исполняются с отношением к текущему местоположению. Как абсолютные или относительные переменные возможно задавание параметров углов и радиусов. Полярное перемещение определяется углом нулевой точки системы в работе и расстоянием от реального расположения инструмента станка.

Исключить преобразование полярных координат в прямоуголные позволяет подготовительная функция

```
G16
G90 G17 G16
G81 G98 X4 Y30 Z-2 R0.7 F45
Y60
Y90
G15 G80
```

Код программы G16 есть модальная функция, исходя из этого она действует до того времени, пока не выключится кодом G15.

G17 – выбор оси XY. Предварительный код G17 необходим для оси XY в качестве настоящей действующей оси. Ось XY становится рабочей при для

исполнения интерполяции в круговом формате, изменения координатной системы и циклов обработки детали в круговой проекции.

G18 – команда выбора плоскости оси XZ. Предварительный код G18 нужен для оси XZ устанавливаемым в параметре текущей оси. Ось XZ устанавливается главной при применении круговой интерполяции, круговом вращении координатной системы и непрерывных циклов обработки деталей.

G19 – команда выбора оси YZ. Предварительная команда G19 необходима для оси YZ устанавливаемой в качестве текущей. Ось YZ устанавливается основной при применении круговой интерполяции и путем постоянных циклов обработки с вращением системы координат.

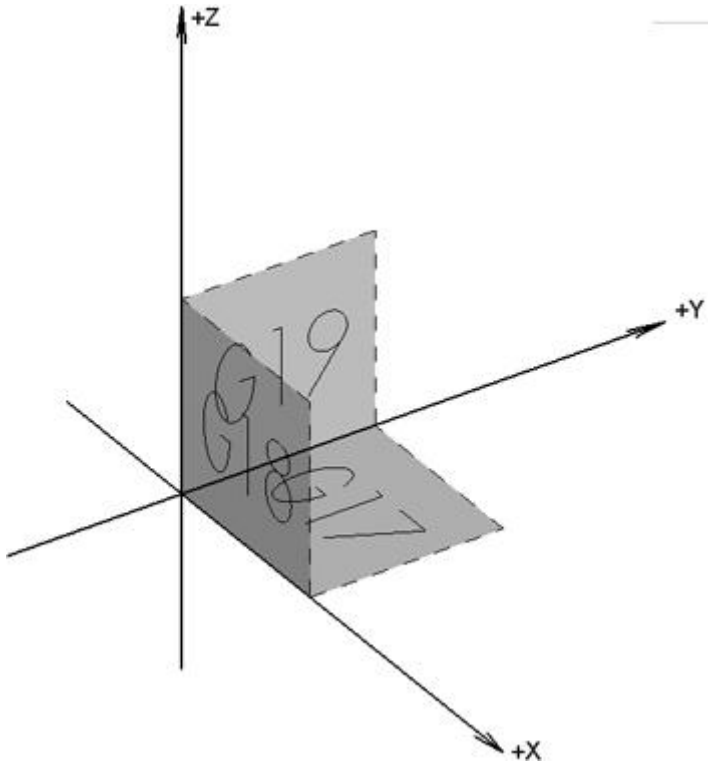

Рис. 16. G19, G18, G17 применяются для активации системы координат в плоскости

G20 – код для ввода параметров в дюймах. Команда G20 использует работу станка с дюймовыми значениями перемещений. Если используется данный параметр, все вносимые параметры устанавливаются как дюймовые значения. Рекомендации во всех циклах, используемых в дюймовых

системах, поставить исполняемую команду G20 в начало кода программы, чтобы в случае, если в программе, исполненной до этого, было включено метрическое вычисление, необходимо задать выбор правильный формат выбора.

Пример:
N10 G20 G40 G49 G54 G80 G90 – код G20 в безопасной строке. Код является модальной функцией и исполняется до поступления кода G21.

G21 – введение параметров в метрической системе координат. Код G21 включает режим работы в метрических параметрах системы. Если включен данный режим, все исполняемые переменные распознаются метрическими. В исполняемых программных кодах, написанных в метрической системе, желательно использовать команду G21 в начале программы, чтобы в случае, когда в программном коде, исполненном до этого, применялось дюймовое вычисление, использовать правильный формат выбора.

Пример:
N10 G21 G40 G49 G54 G80 G90 – команда G21 в безопасной строке. Данный код есть модальная функция и исполняется до команды G20.

G22 – использование условия перемещений до предела. Команда G22 включает предел значений для установки. В данном случае инструмент станка не уходит за область ограниченную пределами. Данные пределы устанавливаются значениями введенными в ЧПУ.

G23 – дезактивация режима предельного позиционирования станка при перемещении. Используя команду G23 заданные параметры перемещения не учитываются. Команда G23 выключает действие команды G22 и осуществляет перемещение инструмента в различные рабочие зоны координатной сетки.

G27 – команда проверки возвращения к стартовым позициям. Команда G27 исполняется также как команда G28. Различие лишь в том, что позиция, в которой переместился рабочий инструмент, не является заданной ранее позицией, тогда случае команды G27 управляющая электроника станка сигнализирует об аварии и выдает аварийный сигнал.

Команды G27 и G28 используются в циклических программных макросах смены инструмента в автоматическом режиме для работы. Предварительно прежде чем исполнить данные G - команды как правило выключают калибровку инструмента станка.

G28 – возвращение позиции при старте в автоматическом режиме. Команда G28 служит для возвращения стартовой позиции станка. То есть быстрое позиционирование рабочего инструмента в позицию условного нуля станка. Возврат к условной нулевой позиции используется для условия параметров проверки и качества обработки материала в программе посередине обработки. Довольно часто команду G28 используют в конечном коде программы управления, для того чтобы по окончании программы, рабочий стол вернулся в позицию, для удобного извлечения обработанного материала.

Для перемещения в исходное положение используется условный кадр следующего вида:

G90 G28 X0.0 Y0.0 Z0.0

В кадре с G28 задаются оси с 0.0 параметрами, перемещение в позицию при старте осуществляется по трем осям. Но не всегда обязательно выполнять данную операцию. Иногда требуется перемещение только по одной из них. Например, для перемещения по осям Z в программном коде для обработки необходимо использовать данный шаг:

G90 G28 Z0.0

Пристальное внимание необходимо обратить на установленный в кадре код G90. Данная команда включает работу станка в относительных координатах. Иначе работает команда G28, она производит программирование некой промежуточной точки, куда будет перемещен инструмент, затем портал возвратится в начальную позицию. Действительно координаты станка, введенные в кадре программы, есть не что иное как координаты промежуточной точки. В данных примерах мы вводили в координаты промежуточной точки с нулевыми значениями. В программном кадре указана команда G90 являющаяся относительной координатой, станок будет перемещаться относительно настоящего положения по каждой оси на ноль миллиметров. Иными словами не будет никуда перемещаться. Вот поэтому при наличии управляющем коде кадра G90 G28 X0.0 Y0.0 Z0.0 портал вернется в начальную позицию без перемещения в промежуточную точку.

При наличии в программе обработки кадра G90 G28 X10.0 Z20.0, портал станка изначально поедет вверх и вправо, а потом переместиться в точку ноль. Команда G28 позволяет осуществить перемещение с ускорением, как и G00, в данном моменте оно возможно будет непрямолинейным. Иными словами портал может что-то «зацепить» . Программист с большим опытом изначально поднимает шпиндель вверх, потом перемещает портал в начальную позицию:

G90 G28 X0.0 Y0.0 Z30.0

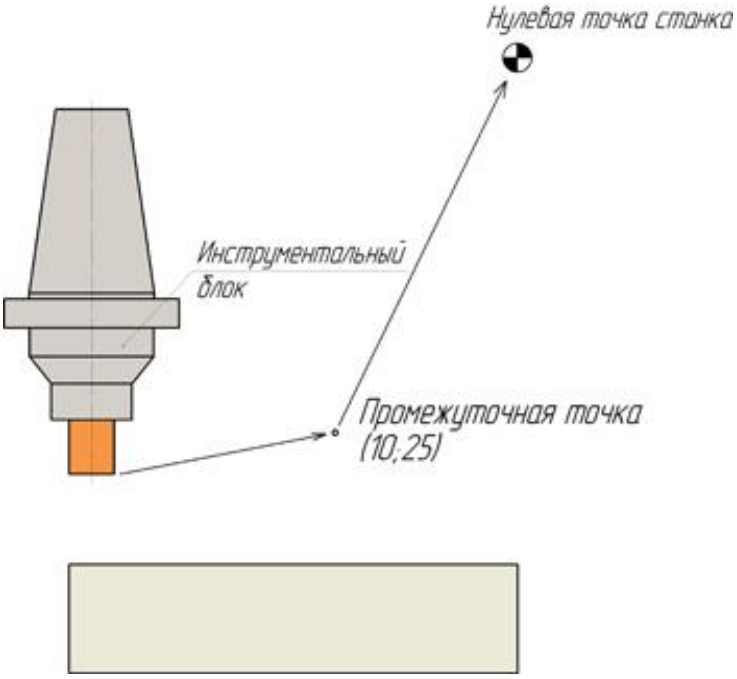

Рис. 17. Когда в программе есть команда G91 G28 X10 Y25, то портал в первую очередь уйдет в промежуточную точку, потом возвратится в начальную позицию.

Не стоит в кадре с G28 использовать команду с абсолютными координатами G91. Если программе имеется кадр G91 G28 X0 Y0 Z0, тогда есть возможность "наезда" рабочего портала на составные части станка или детали.

G30 – команда для возврата в положение смены инструмента. С помощью команды G30 осуществляется перемещение по оси Z к положению для замены инструмента в работе и выключается используемая инструментальная калибровка. Для смены инструмента используется следующая команда подпрограммы:

G30 G90 Z0

Стоит заметить что, если в кадре программы вместо G91 присутствует команда G90, то шпиндель будет перемещаться к поверхности портала.

G31 – команда для пропуска с реакцией на импульс извне. В станках иногда используют команду для пропуска с реакцией на импульс извне. С

помощью немодальной команды G31, оператор выполняет интерполяцию по линии схожую с G01, смешанную с реализацией отклика импульс поступающий извне. Сигнал поступающий от портала подается от нажатия на искомую кнопку панели управления станка, как пример клавиша Старт программы станка.

При отсутствии импульса пропуска, исполняемая программа работает как если был бы задан код G01. Если станок принял поступившую команду, то программа выполняется и переходит к последующей операции.

Рис. 18. Пропуск функцией с реакцией на внешний сигнал

G40 – команда отменяет коррекцию инструмента по радиусу. Коррекция инструмента автоматом по радиусу инструмента выключается путем подачи кодов D00 и G40. Команда G40 стоит в шаге с кодом прямолинейным перемещением с ускорением по контуру детали.

G1 G40 X90

G41 – корректировка радиуса, инструмент располагается слева от детали. Команда G41 применяется для активации коррекции радиуса инструмента в автоматическом режиме, находящегося от детали слева. Направление перемещения сверху вниз, при наблюдении портала сверху, от стороны «+Z» в положение «–Z».

G42 – коррекция радиуса , инструмент располагается с правой стороны от детали. Команда G42 применяется для активации автоматической корректировки радиуса рабочего инструмента, расположенного справа от обрабатываемого объекта. Направление смещения задается, если смотреть сверху вниз, со стороны «+Z» в направлении «–Z».

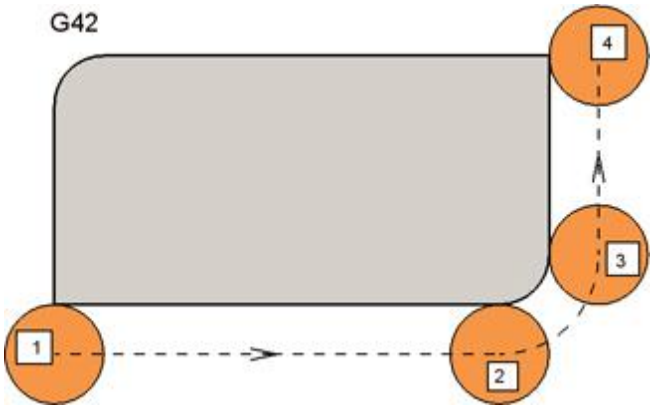

Рис. 19. Коррекция справа

G43 – корректировка длины инструмента. При выполнении управляющей программы начальная позиция рабочего инструмента определяется заданными координатами. Проблема заключается в том, что начальная позиция рабочего механизма не происходит. Обработка происходит кромкой рабочего механизма, которая расположена на определённом отдалении от начальной точки рабочего механизма. Для того чтобы в заданную координату приходил рабочий механизм, необходимо «показать» ЧПУ, на какое расстояние по оси Z нужно сместить эту стартовую точку.

Изменение длины инструмента происходит за счет подачи команды G43 и H-слова информации. Как правило, корректировка расстояние определяется одновременно с ускорением передвижения по оси Z.

Пример: G43 H01 Z100

G49 – дезактивация компенсации расстояния рабочего механизма. Компенсация длины рабочего механизма отменяется путем задания команды G49 или H00.

G50 – дезактивация режима масштабирования. Код G50 предназначен для дезактивации масштабирования G51.

G51 – активация режима масштабирования. В этом режиме программист компенсирует коэффициент масштабирования для координатных осей механизма. Режим включается за счет постоянного кода G51 и отключается G50.

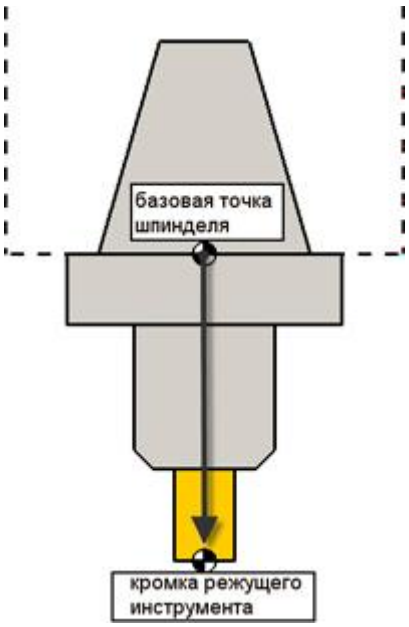

Рис. 20. Команда G43 H изменяет базовую точку рабочего механизма к кромке режущего приспособления

Допускается указание коэффициент масштабирования для всех разом или для определённой оси. Если коэффициент масштабирования больше 1, то система координат увеличивается. Если этот коэффициент масштабирования меньше 1, то координатная система уменьшается.

Для общего изменения масштаба, как правило, применяется следующая форма записи:

G51 X* Y* Z* P*

где G51 – активирует режима масштабирования; X – положение по оси X для промежуточной точки масштабирования; Y – положение по оси Y для промежуточной точки масштабирования; Z – положение по оси Z для промежуточной точки масштабирования; P – коэффициент масштабирования для всех осей вместе.

Возможно также симметричное отображение с помощью отрицательного значения масштабирования. Для отдельного изменения масштаба, как правило, применяется следующий формат:

G51 X* Y* Z* I* J* K*

где G51 – активирует режима масштабирования; X – положение по оси X для промежуточной точки масштабирования; Y – положение по оси Y для промежуточной точки масштабирования; Z – положение по оси Z для промежуточной точки масштабирования; I – коэффициент масштабирования для осей X; J – коэффициент масштабирования для осей Y; K – коэффициент изменения для осей Z.

В функции симметричного отображения применяются между собой отдельное изменение масштабирования и возможность симметричного отображения заданных координат по одним или несколько осям.

Основная программа

G90 G01 F90

M98 P101

G51 X5 Y5 I-1 J1 K1

M98 P101

G51 X5 Y5 I-1 J-1 K1

M98 P101

G51 X5 Y5 I1 J-1 K1

M98 P101

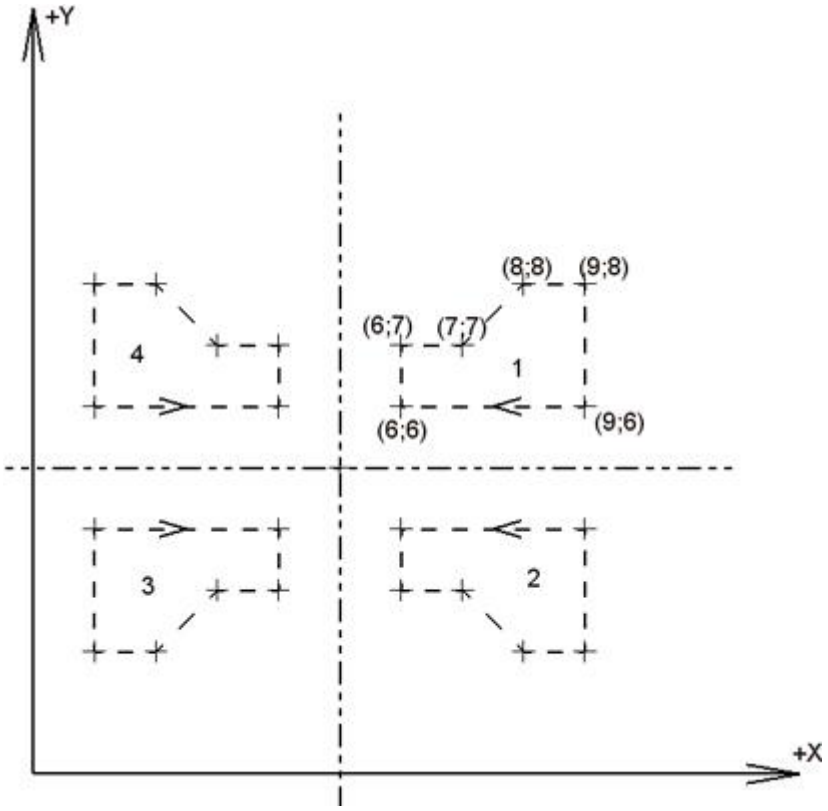

Рис.21. Зеркальное отображение траектории

Подпрограмма

O0101

G90 X6 Y6

Y7

X7

X8 Y8

X9

Y6

X6

M99

G52 – локальная система координат. ЧПУ позволяет задавать, кроме обычных рабочих систем позиционирования, еще и локальные системы позиционирования. Код G52 применяется для нахождения подчиненной

системы позиционирования в диапазоне текущей рабочей координатной системы (G54–G59).

Когда ЧПУ станка применяет команду G52, то начало текущей рабочей системы позиционирования смещается на некое значение, заданное при помощи команд определенных X, Y и Z:

G52 X*Y*Z*

Рис. 22. Локальная система позиционирования

Команда G52 деактивируется, если задается другая рабочая система координат G54–G59 или с помощью команды G52 X0 Y0 Z0

G54–G59 – стандартные действующие системы координат. При помощи команд G54, G55, G56, G57, G58 и G59 задается, в какой действующей системе координат будет выполняться обработка заготовки. Путем выбора различных координатных систем оператор имеет возможность при помощи одних и тех же программ обрабатывать разные заготовки. Если была задействована одна из координатных систем G54–G59, то она применяется до того момента, пока не активируется другая координатная система.

G60 – позиционирование в заданном направлении. При помощи этих команд, ко всем заданным позициям по которым оси можно передвигать из

некоторого направления. Благодаря этому получается возможность убрать ошибки координирования, которые могут появляться из-за неиспользованного хода в системах. Как правило, направление и величина позиционирования указываются параметрами ЧПУ.

G61 – режим очень точного позиционирования останова. Команда G61 используется для включения режима останова. Команда точного останова подробно описана в характеристике команды G09. Единственное отличия между кодой G61 и G09 является то, что G09 немодальная команда. Модальная команда G61 остается включенной до тех пор, пока не будет задана команда на изменение текущего режима, с помощью другой команды, такой как G63.

G64 – режим нарезания. Стандартный режим нарезания активируется кодом G64.

G65 – немодальный вызов микропрограммы. Код G65 позволяет выполнить микропрограммы, расположенная в памяти ЧПУ. Формат для немодального вызова микропрограммы выглядит таким образом:

G65 P_L_

где G65 – команда для вызова микропрограммы; P – номер микропрограммы; L – количество выполнений микропрограммы. Если L не указывается, то ЧПУ принимает, что L = l.

G66 – модальный вызов микропрограммы. Код G66 предназначен для запуска микропрограммы, как и код G65. Единственное различие между этими кодами является в том, что G66 является модальным командой и микропрограммы выполняются при каждом изменении позиции, пока не будет применяться команда G67. Формат для модального активация микропрограммы:

G66 P_L_

где G66 – команда для циклов микропрограммы; P – номер микропрограммы; L – количество циклов микропрограммы.

Если L не указывается, то ЧПУ считает, что L = 1.

G67 – дезактивация модального вызова макропрограммы. При помощи указания кода G67 дезактивируется режим модального вызова макропрограммы G66.

G68 – вращение координатной системы. Модальная команда G68 позволяет выполнить поворот координатной сетки на заданный угол. Для реализации такого вращения требуется указать плоскость вращения, центр поворота и угол вращения. Плоскость поворота устанавливается при помощи кодов G17 (плоскость XY), G18 (плоскость XZ) и G19 (плоскость YZ). Если желаемая плоскость поворота уже активирована, то задание команд G17, G18 и G19 в шаге с G68 нет необходимости.

При действующей программы G90 центр вращения указывается абсолютными координатами относительно стартовой точки станка, если не активирована одна из стандартных рабочих систем позиционирования. Если выбрана одна из рабочих систем позиционирования G54–G59, то центр вращения задается относительно стартовой точки действующей рабочей системы позиционирования. В случае активной команды G91 центр поворота указывается относительно текущего позиционирования. Если же позиции цент вращения не будут заданы, то в качестве позиции центра поворота будет принята текущая позиция.

Угол поворота указывается при помощи R-слова данных. Формат записи для команды разворота координат обычно записывают следующим образом:

G17 G68 X_Y_R_

G69 – отмена разворота координат. При помощи кода G68 отменяется режим поворота позиция.

G73–G89 – постоянные циклы

G80 – Отмена постоянного цикла

G81 – Стандартный цикл сверления

G82 – Сверление с выдержкой

G83 – Цикл прерывистого сверления

G73 – Высокоскоростной цикл прерывистого сверления

G84 – Цикл нарезания резьбы

G74 – Цикл нарезания левой резьбы

G85 – Стандартный цикл обработки

G90 – активация режима абсолютного позиционирования. В этом режиме, абсолютного места положения G90 изменения рабочих механизмов делается относительно нулевой точки механизма или относительно нулевой точки рабочей системы позиционирования G54–G59. Команда G90 является модальным и дезактивируется за счет команды относительного позиционирования G91.

G91 – активация режим относительного местоположения. За счет команды G91 включается режим относительного местоположения. При инкрементном методе отсчета за стартовое местоположение каждый раз используется положение рабочего механизма, в котором он находился перед началом движения к следующей программируемой точке. Команда G91 по сути является модальным и дезактивируется при помощи команды абсолютного позиционирования G90.

G92 – перемещение абсолютной системы координат. Бывают ситуации, когда у оператора ЧПУ возникает необходимость задать определенные переменные в регистрах текущей системы позиционирования для перемещения начальных точек в новое положение.

Код G92 используют для перемещения текущего положения начальной точки за счет редактирования значений в регистрах текущих смещений. Когда ЧПУ выполнит программу G92, то переменные в регистрах смещений изменяются и становятся равными переменным, которые заданы X-, Y- и Z-параметрам.

G92X Y Z

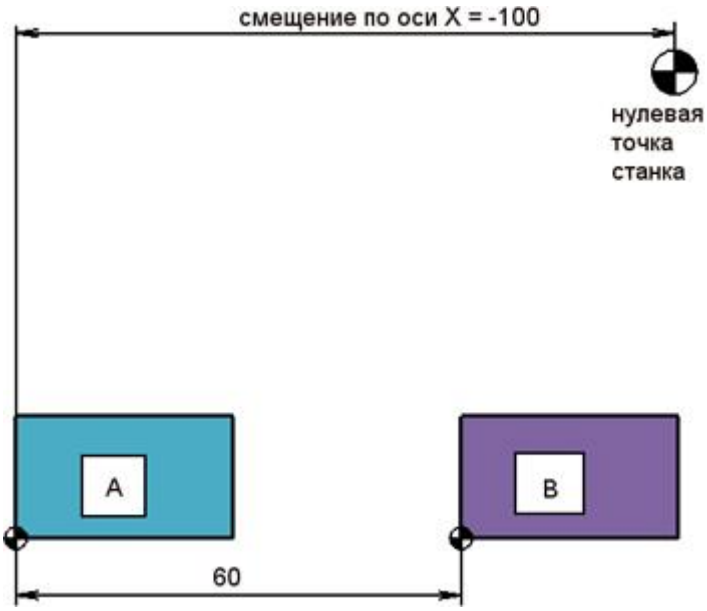

Рис. 23. С помощью G92 мы указываем абсолютной координаты станка и смещаем начальную точку позиционирования

Сначала переместим рабочий механизм в известную нам стартовую начальную точку, а затем применяем G92:

…
G00 X0 Y0
G92 X-70 Y0
…

Кадр G92 X-70 Y0 показывает, что нынешнее текущее положение инструмента определено координатами (–70; 0). Благодаря этому, искомая начальная точка будет позиционироваться на 70 мм правее нынешнего положения рабочего инструмента.

Существует иной метод для достижения этого же результата. Можно сначала переместить рабочий инструмент в позицию, которую надо принять новой нулевой точкой, и затем выполнить команду G92 X0 Y0.

…
G00 X60 Y0
G92 X0 Y0
…

Команда G92 сама по себе не выполняет осевых перемещений. Заданное при помощи G92 перемещение координатной системы на многих станков может быть задано возвратом в стартовую точку или обесточиванием станка.

G94 – скорость движения. При помощи программы G94 заданная скорость перемещения задается в дюймах минуту либо в миллиметрах минуту.

Если действующий дюймовый режим G20, то скорость перемещения F задается как перемещение в дюймах за 60 секунд. Если же в данный момент активен метрический режим G21, то скорость перемещения F принимается как перемещение в миллиметрах за 60 секунд.

G20 F10 – скорость перемещения 10 дюймов в минуту; G21 F10 – скорость перемещения 10 миллиметров в минуту.

Модальный код G94 остается включенным до тех пор, пока не будет выставлена команда G95.

G95 – скорость перемещения механизма в дюймах/миллиметрах на вращение. За счет команды G95 задает скорость перемещения, принимается в дюймах на оборот рабочего механизма или в миллиметрах на оборот рабочего механизма. Другими словами скорость перемещения F синхронизируется со скоростью перемещения рабочего механизма S. При равных значениях F скорость перемещения будет возрастать при увеличении числа подач.

G20 F0.1 – скорость перемещения равна 0.1 дюйма на 1 вращение; G21 F0.1 – скорость перемещения равна 0.1 миллиметра на вращение. Модальная программа G95 остается включенной до тех пор, пока не будет выставлена команда G94.

G98 – возврат к начальной плоскости в программе. Начальная плоскость – это положение по оси Z (уровень), в которой находится рабочий механизм перед активацией текущего цикла. Код G98 дезактивируется при помощи команды G99.

G99 – возвращение к плоскости отвода в цикле. Если цикл работает совместно с кодом G99, то рабочий инструмент перемещается к плоскости возврата посреди всех текущих поверхностей. Плоскость возврата – это позиция по положению Z (уровень), с которой стартует на текущей подаче и в которую перемещается рабочий инструмент после того, как дошел до дна обрабатываемой заготовки. Плоскость возврата обычно задается в цикла при помощью R-адреса. Команда G99 отменяется с помощи программы G98.

Адреса/слова данных

X является программой осевого перемещения. Как правило, за X задаются ось, относительно которой возможно значительно большего перемещение рабочего органа устройства. При этом ось X перпендикулярна к плоскости Z и параллельна рабочего поверхности.

Положительное или отрицательное значение, входящее в состав этого слова, определяет конечную координату исполнительного органа станка вдоль оси X. В кадре можно задать X только 1 раз. Если в 1 кадре будет несколько программ X, то ЧПУ будет работать с последней из них, которая ближе.

Пример:

G01 G90 X100 F30 – линейное перемещение по координате X = 100 со скоростью 30 мм/мин.

Когда X находится в одной позиции с программой выдержки G04, то оно задает время этой выдержки в секундах.

Пример:

G04 X11.0 – выполнить выдержку продолжительностью 11 секунд.

Y является программой осевого перемещения. Ось Y перпендикулярна для 2 осей X и Z. Положительное или отрицательное число, входящее в состав этого цикла данных, задает конечную позицию рабочего органа станка вдоль оси Y. В кадре можно запрограммировать Y только 1 раз. Если в строке будут указаны несколько команд Y, то ЧПУ работает с последней из них, которая значительно ближе к знаку конца кадра.

Пример:

G01 G90 Y20 F220 – линейное перемещение в координату Y = 20 со скоростью 220 мм/мин.

Z служит командой осевого передвижения. Чтобы задать положительного направления Z задают вертикальное направление выводов рабочего инструмента из заготовки. Другими словами ось Z всегда связана со шпинделем станка. Положительная или отрицательная переменная, входящая в состав этого слова данных, задает конечную позицию выполняемого органа станка вдоль оси Z. В кадре можно задать Z только один раз. Если в кадре будут указаны несколько команд Z, то ЧПУ будет работать с последней из них, которая значительно ближе к конца кадра.

Пример:

G01 G90 Z7 F400 – линейное перемещение в позицию Z = 7 со скоростью 400 мм/мин.

A, B, C являются программами иного перемещения. Под круговым передвижением принимают угловое передвижение, поворот оси станка или угловое передвижение управляемого вращением плоскости.

поворотные передвижения рабочего инструмента задают латинскими буквами – A (вокруг X), B (вокруг Y) и C (вокруг Z). Положительные вращения вокруг этих осей определяются довольно-таки просто.

Пример:

G01 G90 C60 F350 – поворот стола на 60° со скоростью 350 мм/мин.

I, J, K задаются во время круговых интерполяции и используются для указании относительных положений от стартовой точки дуги до ее центра. Слово значений с I относится к X, слово данных с J – к Y, а слово данных с K – к Z. При этом в зависимости от местоположения дуги значения могут быть как положительными, так и отрицательными.

R. При текущей круговой интерполяции (G02/G03) R показывает радиус, который соединяет начальную и конечную точки дуги.

Для большинства ЧПУ адрес R может являться командой на выполнение округления при текущей линейной интерполяции. Числовые переменные, входящие в состав R-слова данных, определяет радиус округления.

В постоянных циклах R определяет местоположение плоскости отвода. При использовании с командой поворота координат R задается угол вращения координатной плоскости.

P как правило применяется в постоянных циклах использования и определяет значение времи выдержки на дне изделия. Числовое значение, входящее в состав P-слова данных, как правило, определяет время выдержки в 1 миллисекунду.

Когда P появляется в одном кадре с программой вызова подпрограммы M98, то оно указывает номер запускаемой подпрограммы. В некоторых случаях это же слово значений также может показывать на частоту вызова программы.

Пример:

M98 P1001 – вызов программы O1001.

Q часто применяется в циклах, определяет глубину каждого рабочего хода инструмента.

В цикле Q определяет положение сдвига рабочего инструмента от стенки изготовленного изделия для обеспечения безопасного вывода инструмента из рабочей зоны.

При помощи D задается значение коррекции на радиус рабочего инструмента. Коррекция радиуса рабочего инструмента активируется командами G41 и G42. При помощи программы D00 можно отменить действующую команду коррекции.

При помощи H выбирается значение компенсации длины рабочего инструмента. Компенсация длины рабочего инструмента, как правило активируется командой G43. При помощи команды H00 можно дезактивировать компенсацию длины действующего инструмента.

Для задания скорости подачи служит F-адрес. Если в 1 кадре будут запрограммированы множество скоростей подач, то ЧПУ будет использоваться с последним из них. В случае использования F с командой G94 скорость перемещения будет задаваться в дюймах (G20) либо миллиметрах (G21) в 1 минуту. А в случае использования с G95 скорость подачи будет установлена в дюймах (G20) либо миллиметрах (G21), наоборот. F-адрес именуется модальным, по другому, установленная скорость перемещения остается постоянной до тех пор, пока не новая переменная совместно с F или не изменен режим передвижений при помощи команды G00.

С помощью S применяется число оборотов. S-адрес является модальным, то есть установленное число оборотов остается постоянным до тех пор, пока не заданно иные числовые параметры вместе с S.

При помощи T указывается управление магазином инструментов. Числовое значение с T указывает номер инструмента, который необходимо переместить в позицию смены путем вращения инструментального магазина. Как правило T программируют в одном кадре с командой смены инструмента M06. В этом случае количество значении при T будет определять номер инструмента, который необходимо выбрать из магазина и установить в руку.

Пример:

T3 M06 – задать инструмент № 3.

При помощи N выполняется нумерация кадров УП. При выполнении номера кадра может быть выбран в кадре в любую позицию, но обычно его указывают в начале. Номер кадра не влияет на работу, а помогает программисту ориентироваться в содержании программы.

M-коды

M00 – запрограммированная остановка. Когда ЧПУ применяет команду M00, то выполняется так называемый запрограммированный останов. Все осевые движения останавливаются и стартуют после того, как оператор нажмет кнопку Старт цикла. При этом рабочий механизм продолжает работать и иные функции по-прежнему остаются активными. Если оператор нажимает кнопку Старт, то работа цикла будет продолжено с позиции, следующего за командой M00.

M01 – остановка по выбору. Код M01 служит для остановка по выбору. Работает он по аналогии команде M00, однако оставляет выбор оператору – нужно ли ему или не нужно останавливать работу управляющей. Если эта переключатель нажат, то при чтении команды с M01 происходит остановка. Если переключатель не нажат, то команда M01 игнорируется и выполнение не прерывается.

M02 – конец цикла. Код M02 информирует о завершении программы.

M03 – прямое вращение. При помощи команды M03 запускается прямое (по часовой стрелке) вращение с указанным числом вращений. Команда M03 по-прежнему остается выполняемой до тех пор, пока не будет остановлена командами M04 или M05.

M04 – обратное вращение. При помощи команды M04 запускается обратное (против часовой стрелки) вращение с указанным числом вращений. Код M04 по-прежнему остается выполняемым до тех пор, пока не будет дезактивирована при помощи команд M03 или M05.

M05 – останов. Код M05 прекращает вращение, но не останавливает осевые передвижения.

M06 –смена инструмента. Благодаря коду M06, закрепленный инструмент меняется на следующий, который находится в положении готовности.

M19 – юстировка. При помощи команды M19 осуществляется радиальная юстировка (поворот в определенное положение), чтобы выставить привод на позицию смены инструмента.

M20 – отмена юстировки. При помощи этой команды отменяется команда юстировки шпинделя M19.

M30 – конец программы. Код M30 информирует о завершении цикла.

M98 – вызов сторонней подпрограммы. Команда M98 предназначена для вызова сторонней подпрограммы. Вместе командой используется P-слово данных, оно указывает на номер запускаемой подпрограммы.

Пример:

M98 P1001 – вызвать подпрограмму O1001.

M99 – конец программы. При помощи кода M99 по завершении подпрограммы реализуется возврат к начальной программе, из которой была вызвана данная подпрограмма[4].

1.7 Разработка программы управления ЧПУ станка.

Опираясь на принцип работы ЧПУ станков и особенности работы G-кода, была разработана программа для ARDUINO. Язык программирования устройств Ардуино базируется на C/C++ и скомпонован с библиотекой AVR Libc и позволяет использовать различные ее функции[28]. Вместе с тем он прост в понимании, и на данный момент Arduino — это один из самых удобных способ программирования устройств на микроконтроллерах AVR. Была написана прошивка для микроконтроллера которая считывала траектории с внешнего носителя и передавала команды управления на ДУ ШД, текст этой программы приведен ниже.

```
int motorPins[3][2] = {{8,9},{10,11},{12, 13}};

int count;

int count2[3] = {0,0,0};

int val = 0;

int rot=0;

int incomingByte = 0;

int sign=1;

long delayTime;


//////////////////ручное управление//////////////////


#define VR_X 2         // Ось X подключена к Analog 2

#define VR_Y 1         // Ось Y подключена к Analog 1

int SW1=A0;           // кнопка ручной режим к Analog 0

int LED=A3;           // Индикатор к Analog 3
```

```
int TOG=0;            // Переменная для хранение режима работы пельта
byte value_1, value_2=0;
int value_X, value_Y;  // переменный для хранения осей

//////////////////////////////////////////////////////

//Процедура настройки прошивки
void setup() {

//////////////////ручное управление//////////////////

  pinMode(SW1,INPUT);
  digitalWrite(SW1,HIGH);
  pinMode(LED,OUTPUT);

//////////////////////////////////////////////////////

int i;
Serial.begin(9600);  //Эта скорость должна совпадать со скоростью, установленной в программе

for (i=0; i<3; i++) {
for (count = 0; count < 2; count++) {
```

```
pinMode(motorPins[i][count], OUTPUT); //установка режима работы цифровых pin'ов Ардуино
}
}
delayTime=2000;
}

//Поворот двигателя с номерм sm на один шаг вперёд
void moveForward(int sm) {
digitalWrite(motorPins[sm][1], HIGH); //Задаём направление
digitalWrite(motorPins[sm][0], HIGH);
digitalWrite(motorPins[sm][0], LOW);
}

//Поворот двигателя с номерм sm на один шаг назад
void moveBackward(int sm) {
digitalWrite(motorPins[sm][1], LOW);
digitalWrite(motorPins[sm][0], HIGH);
digitalWrite(motorPins[sm][0], LOW);
}
//Задержка в микосекундах
void delayMicros(long wt){
unsigned long mls;
unsigned int mks;
```

```
mls=(unsigned long)(wt / 1000);
mks=(unsigned int)(wt % 1000);
if (mls>0) delay(mls);
if (mks>0) delayMicroseconds(mks);
}

//Одновременный поворот двигателей 0, 1, 2 на x, y, z шагов соответственно
void MoveSM(long x, long y, long z) {
long c[3], c2[3];
double c1[3], d[3];
long m, i;
boolean flg;

c[0] = x;
c[1] = y;
c[2] = z;

m = 1;
for (i=0; i<3; i++) {
if (m < abs(c[i])) m = abs(c[i]);
}
for (i=0; i<3; i++) {
c1[i] = 0;
d[i] = 1.0 * c[i] / m;
```

```
c2[i] = 0;
}

flg = false;
for (i=0; i<3; i++) {
if (abs(c1[i]) < abs(c[i])) flg=true;
}
while (flg) {
flg=false;
for (i=0; i<3; i++) {
 if (abs(c1[i]) < abs(c[i]))
 c1[i] += d[i];
 if (abs(c1[i]) - abs(c2[i]) >= 0.5) {
 if (c[i]>0) {
 c2[i]++;
 moveForward(i);
 } else {
 c2[i]--;
 moveBackward(i);
 }
 }
 if (abs(c1[i]) < abs(c[i])) flg=true;
}
delayMicros(delayTime);
```

```
}
}
//Основной цикл
void loop() {
//////////////////ручное управление//////////////////
 value_1=digitalRead(SW1);           //дребезг контактов
  if(!value_1)
  {
    delay(50);
    value_2=digitalRead(SW1);
    if(!value_2)
    {
      if(TOG!=0)TOG=0;
      else TOG=1;
      digitalWrite(LED,TOG);        //индикация режима работы
      do{
      }while(!digitalRead(SW1));
      Serial.println("0");
    }
  }
 //////////////Управление с пульта//////////////////
  if (TOG==1)
 {
```

```
  value_X = analogRead(VR_X);                    // Считываем аналоговое
значение оси Y
 if(value_X >= 0 && value_X < 480)
   {
MoveSM(1,0,0);
   }
   if(value_X > 540)
   {
MoveSM(-1,0,0);
   }
       value_Y = analogRead(VR_Y);                    // Считываем аналоговое
значение оси Y
 if(value_Y >= 0 && value_Y < 480)
   {
MoveSM(0,1,0);
   }
   if(value_Y > 540)
   {
MoveSM(0,-1,0);
   }
}
//////////////////////////////////////////////////////
 if (Serial.available() > 0) { //Пришла команда
long c[4]={0,0,0,0};
```

```
int i;
sign=1;
i=0;
incomingByte = Serial.read();
while (incomingByte!=';') { //Читаем входящую строку, признак конца строки знак "точка с запятой"
if (c[i]==0) {
if (incomingByte=='-')
 sign=-1;
}
if (incomingByte==',') {
c[i]*=sign;
sign=1;
i++;
} else if (incomingByte>='0' && incomingByte<='9') {
c[i]=c[i]*10+incomingByte-'0';
}
while (Serial.available() == 0) {
delayMicroseconds(1); //Ждём очередной символ, если не пришел
}
incomingByte = Serial.read();
}
c[i]*=sign;
if (c[3]>0) delayTime=c[3];
```

```
MoveSM(c[0],c[1],c[2]); //Вращаем двигатели на заданное число шагов
Serial.println("OK"); //Отправляем компьютеру сообщение "OK", значит можно высылать новую команду
}
else
delayMicroseconds(1); //Если ничего не пришло, ждём 1 микросекуду.
}
```

Принцип работы данной прошивка заключается в том, что траектории надо писать с указанием количества шагов по осям скорости подачи и необходимости включения или выключения инструмента.

X;Y;Z;S;T.

X,Y,Z –количество шагов; S – скорость; T – включен/выключен инструмент.

Но после проведения серии испытании на реальном оборудовании было выявлены некоторые недостатки этой программы. Заключающий в том, что все траектории будущей продукции необходимо писать полностью вручную и если объект имеет простую форму, то в этом нет никаких сложностей, но для более сложных объектов написание траекторий занимает значительное время и достаточно легко допустить ошибку. Для устранения этого момента необходимо писать специальную программу для компьютера, которая облегчала и автоматизировала бы этот процесс. Но так как уже имеются огромное количество информации по принципу работы G-code, в сети интернета была найдена уже готовая прошивка для Atmega328, в которой уже был реализован частичный функционал G-кода.

Grbl master для работы этой прошивки необходима связь с компьютером который будет передавать инструкции и тексты траекторий на

Atmega328 по сериал порту. Grbl предназначен для трех осевых ЧПУ X, Y, и Z.

G-кодовый переводчик осуществляет подмножество стандарта NIST rs274/ngc. Линейное, круглое и винтовое движение все полностью поддержано.

При работе все параметры сначала записываются в EEPROM, а уже потом обрабатываются контролером для построения траектории движения рабочего механизма, что значительно повышает качество готового изделия тк минимизируется вероятность пропуска шага из за некачественного информационного провода по которому передаются инструкции.

Grbl поддерживает все общие операции, с которыми сталкиваются в производстве, но некоторые функции небыли реализованы такие как: переменные, базы данных инструмента, контроля цикла циклы, арифметика и структурный контроль. Реализованы только основные машинные операции и возможности.

Но т.к цель работы сделать полноценный пульт управления CNC станка то необходимо избавится от использования в процессе работы компьютера и для решения этой задачи была использована ещё 1 плата Arduino Nano цель которой заменить компьютер. Для этой платы была разработана прошивка в которой реализованы функции ручного управления 3 осями, выход в стартовую позицию, считывания инструкции и траекторий с SD карты на которую те предварительно были загружены с компьютера. Текст прошивки представлен ниже.

```
#include <SD.h>

#include <SPI.h>

const int chipSelect = 4;

const int butPinX = 5;
```

```
const int butPin_X = 6;
const int butPinY = 7;
const int butPin_Y = 8;
const int butPin_0 = 9;
int valueX, value_X, valueY, value_Y, value_0;
int TOG=0;
int t;
// int s;
int pos;
int value;
int incomingByte=0;
void setup()
{
  Serial.begin(115200);
  pinMode(butPinX, INPUT);
  digitalWrite(butPinX, HIGH);
  pinMode(butPin_X, INPUT);
  digitalWrite(butPin_X, HIGH);
  pinMode(butPinY, INPUT);
  digitalWrite(butPinY, HIGH);
  pinMode(butPin_Y, INPUT);
  digitalWrite(butPin_Y, HIGH);
  pinMode(butPin_0, INPUT);
  digitalWrite(butPin_0, HIGH);
```

```
  !SD.begin(chipSelect);
}
void loop()
{

//////////////////ручное управление X//////////////////

  if(!digitalRead(butPinX))
  {
    delay(50);
    if(!digitalRead(butPinX))
    {
      Serial.println("G91 G0  X1");
      do{
      }while(!digitalRead(butPinX));
      }
     }

//////////////////ручное управление _X//////////////////

  if(!digitalRead(butPin_X))
  {
    delay(50);
    if(!digitalRead(butPin_X))
```

```
    {
      Serial.println("G91 G0  X-1");
      do{
      }while(!digitalRead(butPin_X));
      }
     }

//////////////////ручное управление Y//////////////////

  if(!digitalRead(butPinY))
  {
    delay(50);
    if(!digitalRead(butPinY))
    {
      Serial.println("G91 G0  Y1");
      do{
      }while(!digitalRead(butPinY));
      }
     }

//////////////////ручное управление _Y//////////////////

  if(!digitalRead(butPin_Y))
  {
```

```
   delay(50);
   if(!digitalRead(butPin_Y))
   {
     Serial.println("G91 G0  Y-1");
     do{
     }while(!digitalRead(butPin_Y));
     }
    }

//////////////////ручное управление _0//////////////////

  if(!digitalRead(butPin_0))
  {
    delay(50);
    if(!digitalRead(butPin_0))
    {
      Serial.println("G90 G28 X0 Y0");
      do{
      }while(!digitalRead(butPin_0));
      }
     }
if  (t==10)
{
t=0;
```

```
Serial.print("?");
}
t++;
  incomingByte = Serial.read();
if (incomingByte=='I')
   {
    File dataFile = SD.open("GCODE.TXT", FILE_READ);
    if (dataFile)
      {
       while  (dataFile.available() && value!='G' && value!='M')
         {
          if (value=='g')
         {
          Serial.write('G');
         }
          if (value=='m')
         {
          Serial.write('M');
         }
          dataFile.seek(pos);
          value = dataFile.read();
          if (value!='G' && value!='M')

             {
```

```
            Serial.write(value);
          }


        pos ++;
        delay (3);
       }
     dataFile.close();
   }
      if (value=='G')
      {
       value='g';
      }
       if (value=='M')
      {
       value='m';
      }
  }
}[29][30]
```

1.8 Драйвер для управление шаговым двигателем

Существует огромное разнообразие шаговых двигателей различных размеров и характеристик. Чтобы управлять ими нужно специальное устройство — драйвер. Для работы драйвера необходимо подавать 3 сигнала

Enable - разрешение на вращение

Step - количество шагов

Dir - направление вращения

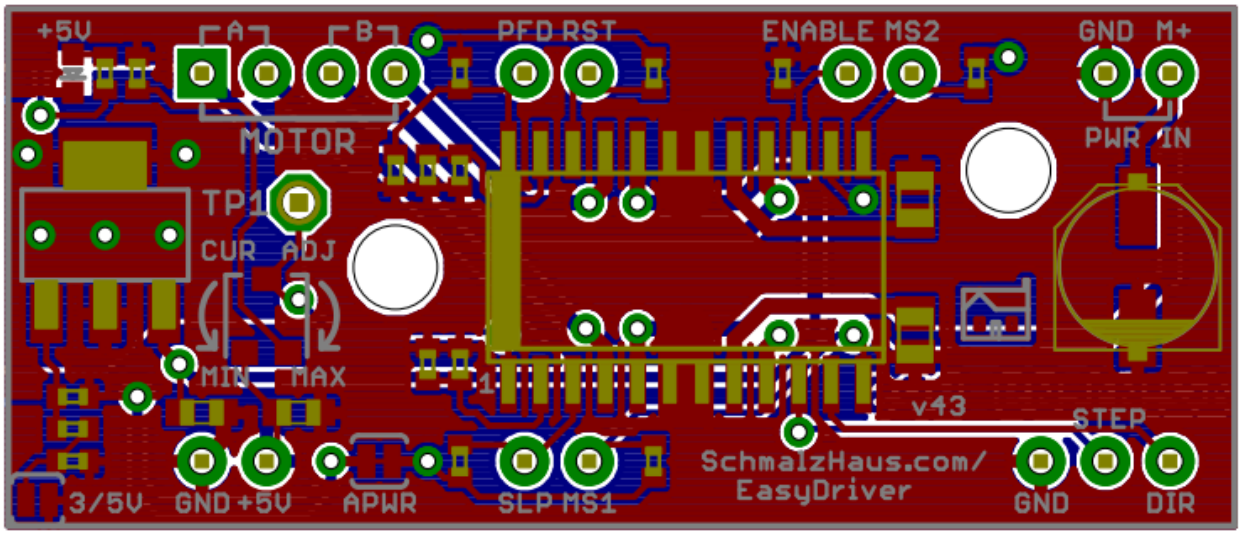

Рис.24. Драйвер шагового двигателя

Для разрабатываемого лазерного гравера подойдет недорогой компактный драйвер компании A4988 имеющий следующие характеристики:

- модель: A4988;
- напряжения питания: от 7 до 34 В;
- возможность установки шага: от 1 до 1/16 шага;
- напряжение логики: 5 В;
- защита от перегрева;
- максимальный ток на фазу: 2 А с радиатором;

- расстояние между рядами ножек: 11 мм;
- размер платы: 21 х 16 мм;
- габариты драйвера: 21 х 16 х 15 мм;
- габариты радиатора: 10 х 6 х 10 мм;
- вес с радиатором: 4 г;

Плата создана на базе микросхемы A4988 компании - драйвера биполярного шагового двигателя. Особенностями A4988 являются регулируемый ток, защита от перегрузки и перегрева, драйвер также имеет пять вариантов микрошага (вплоть до 1/16-шага). Он работает от напряжения 7 - 34 В и может обеспечить ток до 2 А на каждую обмотку.

Описание:

Этот драйвер позволит управлять биполярным шаговым двигателем с выходным током до 2 А на обмотку (для получения дополнительной информации смотрите раздел о рассеивании мощности). Ниже приведены ключевые особенности драйвера:

- Простой интерфейс управления шагом и направлением вращения электродвигателя
- Пять различных разрешений перемещения: полный шаг, 1/2-шага, 1/4-шага, 1/8-шага, 1/16-шага
- Регулируемый контроль тока с помощью потенциометра, позволит установить максимальный выходной ток. Это даст вам возможность использовать напряжение выше допустимого диапазона для достижения более высокой угловой скорости шага двигателя
- Интеллектуальное управление автоматически выбирает режим регулировки затухания тока (медленный и быстрый режимы)
- Защитное отключение при перегреве и перегрузке по току, а также блокировка питания при пониженном напряжении

- Защита от короткого замыкания на землю, защита от замыкания в нагрузке

Использование:

Соединение с источником питания:

Для работы с драйвером необходимо питание логического уровня (3 - 5,5 В), подаваемое на выводы VDD и GND, а также питание двигателя (8 - 35 В) на выводы VMOT и GND. Чтобы обеспечить необходимый потребляемый ток (при пиковых до 4 А), необходимо поставить конденсаторы для гальванической развязки как можно ближе к плате.

Размер шага (и микрошага):

У шаговых двигателей обычно установлена конкретная величина (например 1,8° или 200 шагов на оборот), при которой достигается полный оборот в 360°. Микрошаговый драйвер, такой как A4988 позволяет увеличить разрешение за счёт возможности управления промежуточными шагами. Это достигается путём возбуждения обмоток средней величины тока. Например, управление мотором в режиме четверти шага даст двигателю с величиной 200-шагов-за-оборот уже 800 микрошагов при использовании разных уровней тока.

Разрешение (размер шага) задаётся комбинациями переключателей на входах (MS1, MS2, и MS3). С их помощью можно выбрать пять различных шагов, в соответствии с таблицей ниже. На входы MS1 и MS3 переключателя установлены 100 кОм подтягивающие на землю резисторы, а на MS2 - 50 кОм, и если оставить их не подключёнными, двигатель будет работать в полношаговом режиме. Для правильной работы в режиме микрошага необходим слабый ток, который обеспечивается ограничителями по току. В противном случае, промежуточные уровни будут некорректно восприниматься, и двигатель будет пропускать микрошаги.

Таблица 3. Настройки шага управления ДУ

MS1	**MS2**	**MS3**	**Разрешение микрошага**
Низкий	Низкий	Низкий	Полный шаг
Высокий	Низкий	Низкий	1/2 шага
Низкий	Высокий	Низкий	1/4 шага
Высокий	Высокий	Низкий	1/8 шага
Высокий	Высокий	Высокий	1/16 шага

Входы управления:

Каждый импульс на входе STEP соответствует одному микрошагу двигателя, направление вращения которого зависит от сигнала на выводе DIR. Обратите внимание, что выводы STEP и DIR не подтянуты к какому-либо конкретному внутреннему напряжению, поэтому вы не должны оставлять эти выводы плавающими при создании приложений. Если вы просто хотите вращать двигатель в одном направлении, вы можете соединить DIR непосредственно с VCC или GND. Чип имеет три различных входа для управления состоянием питания: RST, SLP и EN. Обратите внимание, что вывод RST плавает; если вы его не используете, вы можете подключить его к соседнему контакту SLP на печатной плате, чтобы подать на него высокий уровень и включить плату.

Ограничение тока:

Для достижения высокой скорости шага, питания двигателя, как правило, гораздо выше, чем это было бы допустимо без активного ограничения тока. Например, типичный шаговый двигатель может иметь максимальный ток 1А с 5 Ом; сопротивлением обмотки, отсюда максимально допустимое питание двигателя равно 5 В (U=I*R). Использование же такого двигателя с питанием 12 В позволит повысить скорость шага. Однако чтобы предотвратить повреждение двигателя, необходимо ограничить ток до уровня ниже 1 А.

A4988 поддерживает активное ограничение тока, которое можно установить подстроечным потенциометром на плате. Один из способов

установить предельный ток - подключить драйвер в полношаговый режим и измерять ток, протекающий через одну обмотку двигателя без синхронизации по входу STEP. Измеренный ток будет равен 0,7 части предельного тока (так как обе обмотки всегда ограничиваются примерно на 70% от текущей настройки предельного тока в полношаговом режиме). Учтите, что при изменении логического напряжения Vdd, на другое значение, изменит предельный ток, поскольку напряжение на выводе "ref" является функцией Vdd.

Еще один способ установить предельный ток – измерить напряжение на выводе "ref" и вычислить полученное ограничение тока (резисторы SENSE равны 0,05 Ом). Напряжение вывода доступно через металлизированное сквозное отверстие (в кружке на шёлкографии печатной платы). Ограничение тока относится к опорному напряжению следующим образом:

Current Limit = VREF × 2,5

Например: опорное напряжение равно 0,3 В, предельный ток 0,75 А. Как упоминалось выше, в режиме полного шага, ток через катушки ограничен 70% от текущего предела, поэтому, чтобы получить полный шаг тока катушки в 1 А, текущий предел должен быть 1 А / 0,7 = 1,4 А, что соответствует VREF 1,4 А / 2,5 = 0,56 В. Смотрите спецификацию A4988 для получения дополнительных сведений.

Рекомендации по рассеиванию мощности:

Максимально допустимый ток подаваемый на обмотку, у микросхемы A4988 равен 2 А. Фактический ток, который можно подать на плату, зависит от качества охлаждения микросхемы. Плата разработана с учётом отвода тепла от микросхемы, но при токе выше 1 А на обмотку необходим теплоотвод или другое дополнительное охлаждение.

Обратите внимание, что ток, измеренный на источнике питания, как правило, не соответствует величине тока на обмотке. Так как напряжение,

подаваемое на драйвер, может быть значительно выше напряжения на обмотке, то, соответственно, измеряемый ток на источнике питания может быть немного ниже, чем ток на обмотке (драйвер и обмотка в основном работают в качестве переключаемого источника с пошаговым понижением питания). Кроме того, если напряжение питания намного выше необходимого двигателю уровня для достижения требуемого тока, то скважность будет очень низкой, что также приводит к существенным различиям между средним и RMS током (среднеквадратичное значение переменного тока).

Время переключения драйвера из режима Disabled в режим Enabled составляет 15 миллисекунд. Таким образом, после подачи на вход ENBL драйвера A4988 сигнала, включающего управление, необходимо сделать паузу 20 миллисекунд, и только после этого посылать управляющие сигналы на PUL.

- Если контролировать включение-выключение управления шаговым двигателем, то на вход ENBL драйвера A4988 можно подать +5В от выхода Ардуино. Также, если вообще ничего не подключать к входу ENBL драйвера TB6560, то он будет в состоянии включен (Enable).

- В режиме 200 шагов на оборот между пульсациями на входе PUL необходимо сделать паузу 2 миллисекунды, чтобы дать шаговому двигателю отреагировать на команду перемещения ротора. То есть, если на PUL передать следующие сигналы:

HIGH - LOW - [пауза 2мс] – HIGH – LOW - [пауза 2мс] – HIGH – LOW - [пауза 2мс], то ротор ШД сделает 3 шага за 6 миллисекунд.

1.9 Разработка схемы управления

Для работы лазерного гравера была разработана схема подключения всех необходимых элементов, двух драйверов управления шаговых двигателей оси X и оси Y, а также 2 микроконтроллеров в одной будет прошивка grbl-master которая преобразует G-code в машинный для управления драйверов, а в другой программа для чтения с SD карты траекторий и передачи их в первый(рис.1).

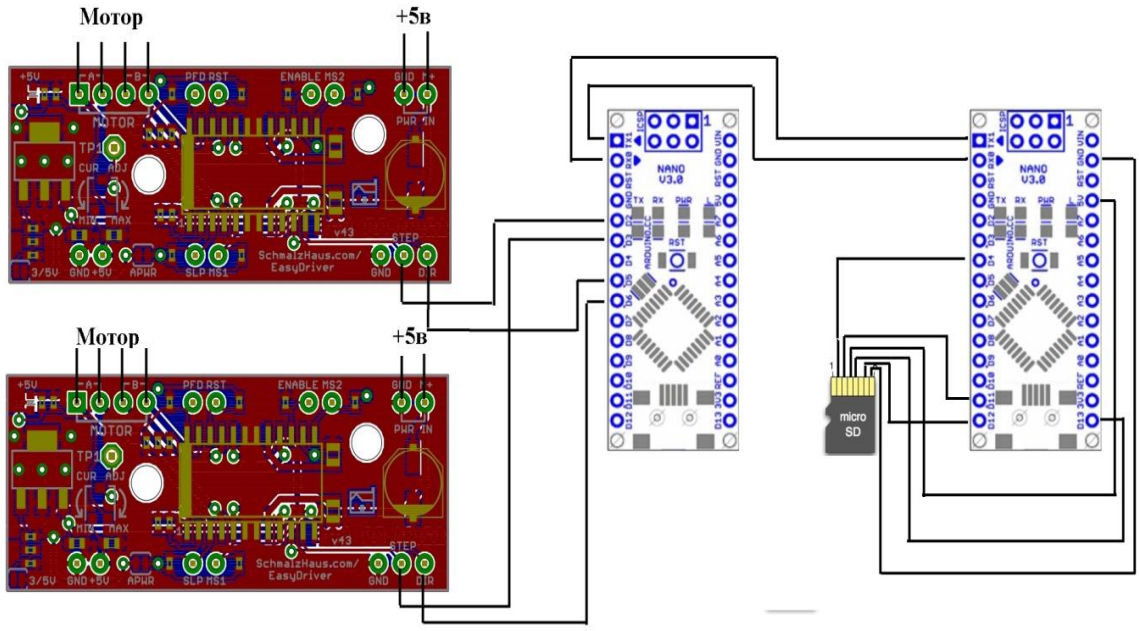

Рис.25.Схема подключения автономного лазерного гравера.

Табл.4. Подключение SD карты

micro SD card	Контакт ATMEGA328
2	D4
3	D11
4	5v
5	D13
7	D12
8	GND

Табл.5. Подключение микроконтроллеров к драйверов ШД

1й МК atmega328	2й МК atmega328	1й Драйвер ШД	2й Драйвер ШД
tx	rx		
rx	tx		
	D2	dir	
	D3		dir
	D5	step	
	D6		step

1.10 Экономическая эффективность

Расчет себестоимости

Полную стоимость комплектующих изделий определили по таблице 6

Таблица 6. Стоимость компонентов

Наименование изделия	Марка, размер	ГОСТ, ТУ	Количество, шт	Цена за единицу, (руб.)	Затраты, (руб.)
1	2	3	4	5	6
Микросхемы	A3967 driver chip	ГОСТ17467-79	3	50	150
	Atmega328		2	100	200
	LM317MDCYR		3	15	45
	CH340		2	20	40
Диоды, стабилитроны, светодиоды, оптопары	LED RED 591NM 0603 SMD (2mA, 1.8V)	ТУ 362.029	2	10	20
	LED YELLOW 591NM 0603 SMD (2mA, 1.8V)		3	11	33
	LED BLUE 591NM 0603 SMD (2mA, 1.8V)		2	15	30
	LED GREEN 591NM 0603 SMD (2mA, 1.8V)		2	12	24

Продолжение таблицы 6

Наименование изделия	Марка, размер	ГОСТ, ТУ	Количество, шт	Цена за единицу, (руб.)	Затраты, (руб.)
1	2	3	4	5	6
	1N4004		2	2	4
	1N4148		2	2	4
	MBR0520		4	2	8
Конденсаторы	CAP 100UF 35V ELECT MZA SMD	ОЖО.464. 214ТУ	3	20	60
	CAP 1UF 10V CER Y5V SMD0603		3	5	15
	CAP 680PF 50V CERAMIC X7R 0603		3	5	15
	22пФ		3	5	15
	100нФ		4	5	20
	CAP CER .10UF 50V Y5V 0603		8	5	40

Продолжение таблицы 6

Наименование Изделия	Марка, размер	ГОСТ, ТУ	Количество, шт	Цена за единицу, (руб.)	Затраты, (руб.)
1	2	3	4	5	6
Резисторы	RES 20K OHM 1/10W 5% 0603 SMD	ГОСТ 7113-77	3	2	6
	RES .75 OHM 1/4W 1% 0805 SMD		3	2	6
	RES 5.1K OHM 1/10W 5% 0603 SMD		3	2	6
	RES 10K OHM 1/10W 5% 0603 SMD		13	2	26
	RES 1.0K OHM 1/10W 5% 0603 SMD		13	2	26
	RES 240 OHM 1/10W 5% 0603 SMD		3	2	6

Продолжение таблицы 6

Наименование Изделия	Марка, размер	ГОСТ, ТУ	Количество, шт	Цена за единицу, (руб.)	Затраты, (руб.)
1	2	3	4	5	6
	RES 715 OHM 1/10W 1% 0603 SMD		3	2	6
	RES 910 OHM 1/10W 5% 0603 SMD		3	2	6
	RES 4.7K OHM 1/10W 1% 0603 SMD		2	2	4
Стеклотекстолит	СФ-2-35		1	120	120
Кабель экранированный	КГВЭВ 4х0,5		1	50	50
Корпус	BOX-103		1	70	70
Припой	Sn60/ Pb40		1	50	50
Флюс	ЛТИ-120		1	30	30
Итого					1224

Стоимость покупных комплектующих изделий с учетом транспортно-заготовительных расходов:

$$C_{покуп} = \sum_{i=1}^{n} S_{покуп} \cdot (1 + K_{тз}),$$

где $К_{ТЗ}$ – коэффициент транспортно-заготовительных расходов; Ктз =0,04

$S_{покуп}$ – стоимость покупных комплектующих изделий, руб.

$С_{покуп}$=1224х(1+0,04)=1272,96 руб.

Цены актуальны на 22.05.2016г. Валютный курс составлял 65 рублей 41 копейка за 1 доллар. Закупка проводилась в магазинах:

http:\\www. chipidip.ru http:\\www.ebay.com

1.11 Изготовление печатной платы

Конструкторско- технологический раздел.

Составили принципиальную схему станка с числовым программным управлением. Соответственно схеме принципиальной сделали разводку печатной платы для разрабатываемого устройства.

В станке применяется двухсторонняя печатная плата выполненная на одностороннем стеклотекстолите. Токопроводящие слои платы делаются комбинированным методом. Проводники изготавливаются методом травлением фольгированной подложки, а отверстия с металлизацией - методом электрохимии. Вывод элемента и крепление проводников на печатной плате делают с помощью контактной площадки выполненной в утолщенным участком с большей шириной.

Размеры станка, его конфигурация и фиксирование печатной платы делаются исходя из его расчетных размеров, пайки, элементной базы, контролируемых данных и технических и экономических показателей.

Внешние размеры платы печатной принимаются с выбором класса точности, обязательно учитывается то, что с укрупнением размеров, будут расти погрешности, допуск при исполнении конструктивных элементов для фиксации их на печатной плате. Изготавливаемая печатная плата имеет форму в виде простого прямоугольника.

Станок представляет собой конструкцию с печатной платой размерами 200 х 150 мм с размещенными на ней электронными деталями и контактами для присоединения источника питания и силовых цепей.

Для конструирования электрохимическим методом печатных плат берется листовой материал с изоляционным основанием и нанесенной на него металлической фольгой с двух сторон. В данной магистерской работе выбираем стеклотекстолит фольгированный марки СФ-2-35. Нужными

параметрами номинальных толщин двухсторонних печатных плат являются: 0,5; 1; 1,5; 2; 2,5; 3 (мм). Берем значения $H_т$ равное 2 мм.

Монтажное отверстие находится в зоне контактной площадки, в которое будет вставляться вывод элемента схемы. Монтажное отверстие имеет металлизированные боковые стенки. Применение отверстий с металлизацией необходимо для большей надежности печатной платы.

Диаметр самих отверстий сделанных в печатной плате должен быть больше диаметра, вставляемого в него вывода элемента, это дает возможность беспрепятственного монтажа устанавливаемого элемента на плату. Размер отверстия с металлизацией влияет от толщины используемой платы. Это связано с тем, что при осаждении металла с помощью гальваники на стенках малого размера, выполненного с толстым основанием, неравномерной получается толщина слоя металлизации, непокрытыми могут остаться места с большим соотношением длины к диаметру отверстий. Размер отверстия с металлизацией необходимо выбирать с условием, что он будет составлять половину от толщины платы.

Размер допустимой минимальной ширины проводника на печатной плате выбираем равное 1мм, минимальное расстояние в пределах допуска между элементами выбираем равное 1,5мм.

Проводники на печатной плате изготавливаются с одинаковой шириной во всем диапазоне разводки. Сильноточные цепи силовых элементов, и управления, выполняются по другому принципу, исходя из нагрузки на их токопроводящие цепи.

Печатные проводники размещаются равномерным монтажом по всей поверхности печатной платы, учитываются следующие требования: координатная сетка должна быть параллельной или под углом, кратным 15°; направление движения волны припоя должно быть параллельно угол к нему не должен превышать более 30° со стороны пайки элементов, при условии

что проводящий рисунок не покрыт защитной маской со стороны пайки; к контуру проводящей площади он должен находится перпендикулярно касательной.

Расположение электрокомпонентов на печатной плате выбирается согласно печатному узлу и его конструктивным требованиям. При размещении электрокомпонентов учитываем: их взаимное расположение, рациональную компоновку, дающую более логичную трассировку и убирающее паразитное влияние на слаботочные элементы управления; обеспечить основные технологические требования, предъявляемые к станкам (автоматизированная сборка и пайка, контроль узлов); данными мерами обеспечить повышенную надежность, выполнение высоких массогабаритных показателей, обеспечить высокое быстродействие, выбрать оптимальный температурный режим с применением теплоотвода, предусмотреть возможность хорошей ремонтопригодности.

Края печатной платы изготавливаются параллельно в соответствии с линиями координатной сетки устройства.

Отверстия и электрокомпоненты токопроводящего медного слоя размещаются на печатной плате в соответствии с базой координат сетки. Электрокомпоненты токопроводящего медного слоя размещаются с краю печатной платы на расстоянии больше чем толщина самой платы с учетом допусков и погрешностей при проектировании и учетом линейных размеров печатной платы.

Координатная сетка наносится с шагом 2,54 мм непосредственно на чертеж. Центры отверстий для монтажа располагаются в компонентах сетки координат. При установке на печатную плату электрокомпонента имеющего два вывода и более , при условии что отверстия между его выводами кратно координатной сетке устройства, элемент размещаем выводами в отверстия на узлах сетки устройства. Размещаемый электрокомпонент который не имеет

выводов, размер между которыми кратный шагу координатной сетки устройства, то в узле координатной сетки располагается один вывод, а центр отверстия на горизонтальной и вертикальной линиях координатной сетки под другой вывод.

Изготовление печатных плат - это очень сложный технологический процесс. Рассчитанная конструкция печатной платы устройства и ее чертежи отправляются в лабораторию, там изготавливаются фотообразцы на твердом носителе.

В лаборатории размещены устройства, которые позволяют масштабировать размер платы в несколько раз.

Основания плат в последствии перевозят в цех специальной обработки, там они доводятся до необходимых размеров в соответствии с техзаданием. Выполняется сверление отверстий и разные операции с материалами, электрокомпонентами и основаниями печатных плат.

Как только печатная плата прошла техническую обработку основания платы попадает в серию на производстве, автоматизированными линиями происходит металлизация отверстий, на отверстия осаживается медь. Затем плата лудится и покрывается свинцом и оловом, в последствии происходит травление. В закрытых емкостях с печатной платы снимается поверхностный слой, на печатной плате остается лишь металл - токопроводящие дорожки.

Затем эта печатная плата с токопроводящими дорожками отправляется на отделение для прессовки, там выполняются работы на прессе согласно установочным размерам печатной платы.

Предварительно обработанные печатные платы отправляются на участок оксидирования, в последствии поступают на участок фрезерования, где происходит их обработка по контуру в соответствии с расчетными размерами.

После полной обработки основания печатной платы происходит установка электрокомпонентов. Электрокомпоненты на плате размещаются как вручную, или с применением "лазерного щупа": основание печатной платы размещается в специальном пазу монтажного стола, лазерным лучом отмечаются места на печатной плате, где будут размещаться электрокомпоненты одного и того же типа.

Пайка электрокомпонентов будет происходить волной припоя, на участок с жидкой расплавленной смесью свинца и олова погружается печатная плата с установленными элелектрокомпонентами и при определенной температуре, регулируемой специальными автоматами, в отверстиях печатной платы выполняется пайка выводов электрокомпонентов.

Для защиты печатной платы от внешних вредных воздействий окружающей среды и других агрессивных сред, наносится защитная маска. Печатная плата маркируется и передается на контроль в технический отдел ОТК.

1.12 Экспериментальные исследования

После всех работ был собран опытный образец (рис.1) для проверки в работе. При постройки использовались два ДВД привода, деревянная доска 45x20x2000 и саморезы. Лазер был взят из самого ДВД привода и помещен в корпус от лазерной указки для фокусировки луча, выравнивание углов производилось при помощи строительного уголка. В качестве рабочей платформы использовался текстолит к которому прикреплялась обрабатываемая заготовка. Электрическая часть была изготовлена по технологии ЛУТ и спрятана под осью Y, в качестве источника питания использовалась зарядка от сотового телефона 5вольт 2 ампера.

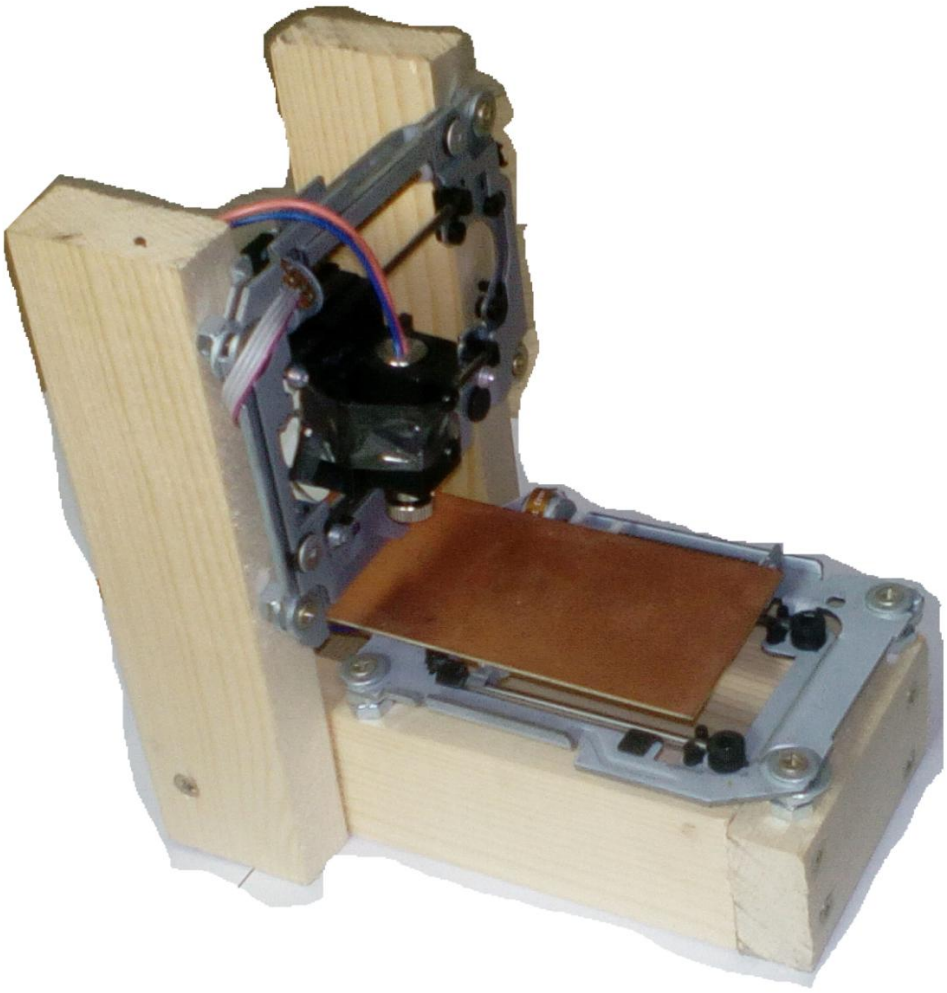

Рис .26. Опытный образец автономного лазерного гравера.

Программа для рисования и создания векторной графики и преобразования её в G-code использовалась Inkscape

Inkscape - свободно распространяемый векторный графический редактор, удобен для создания как художественных, так и технических иллюстраций. Это стало возможным во многом благодаря открытому формату SVG, развиваемому консорциумом W3C. Формат SVG позволяет создавать иллюстрации различного типа, в том числе анимированные. Поскольку SVG основан на языке разметки XML, к нему можно писать расширения, чем авторы Inkscape и пользуются. Программа распространяется на условиях GNU General Public License.

После чего полученные файлы были перенесены на SD карту и выгравированы на картоне рис.31.

Рис 27. Гравировка на картоне построчная

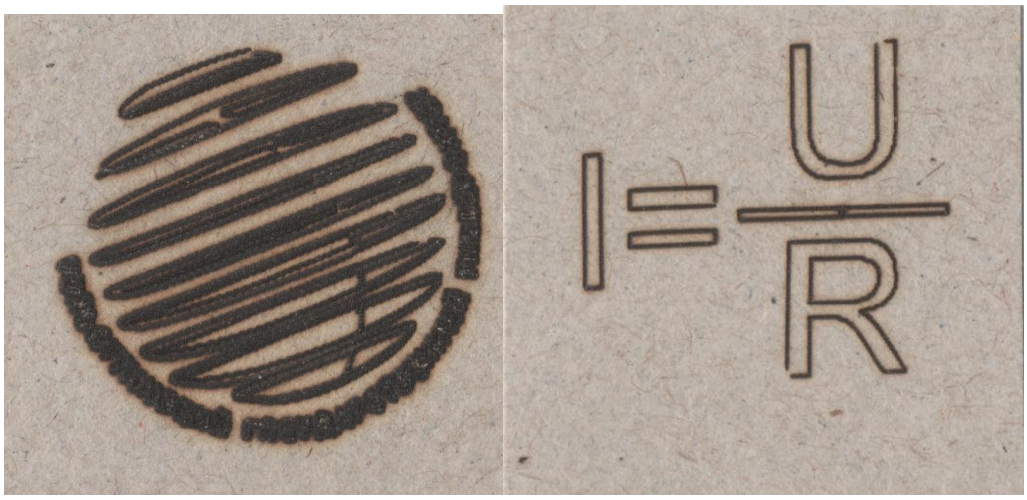

Рис 28. Гравировка на картоне построчная

Как видно из рисунков данный лазерный гравер можно использовать для создания всевозможных рисунков используя построчное выжигание, и выставление скорости обработки отдельных частей рисунка рисунок 27 размером 20х20 выжигался 34 минуты. Также установив достаточно мощный лазер на нем, можно резать некоторые материалы, тк станок поддерживает векторную графику и умеет выключать и включать лазер, а также обводить контуры рисунка, что видно на рисунке 28.

ЗАКЛЮЧЕНИЕ

В результате проведенных исследований и расчетов было спроектирован пульт управления для станок с числовым программным управлением. Согласно техническому заданию данное устройство способно выполнять считывание рабочих программ с внешних носителей и преобразовывать их в машинные коды для управления рабочим механизмом станка. Это подтверждено практической постройкой системы. В ходе проектирования был использован микроконтроллер архитектуры ARM, что позволило эффективно использовать вычислительную мощность микроконтролера.

Список используемой литературы и источников

1. Семенкова О. И. Автоматизация проектно-конструкторских работ и технологической подготовки производства в машиностроении. Т. 1/Под ред. - Минск: Высшая школа, 2005, с. 352

2. Фролова К. В. Механика промышленных роботов: Учеб. Пособие для вузов: Под ред. - Минск: Высшая школа, 1988, с. 304

3. Гольдштейн А.И. Молочник В.И. О внутренней структуре постпроцессоров. — В кн.: Повышение эффективности использования станков с ЧПУ: Под ред. - Киев: Знание, 2006, с. 26

4. Ловыгин А. А. Теверовский Л. В. Современный станок с ЧПУ и С AD & CAM система: Под ред. - МДК, 2012, с. 279

5. Старков В. К. Обработка резанием. Управление стабильностью и качеством в автоматизированном производстве/ М.: Машиностроение, 1989, с. 296

6. Косиловой А.Г. Справочник технолога-машиностроителя: в 2 т. Т. 2 / под ред., Р.К. Мещерякова. 4-е изд., перераб. и доп. М.: Машиностроение, 1985, с. 496

7. Патент № 2063307 Россия, С1 В 23 В 25/06. Способ определения допустимой скорости резания при механической обработке детали твердосплавным инструментом / А.Л. Плотников. № 94010673/08; Заявлено 29.03.1994; Опубл. Бюл. № 19, 1996.

8. Старков В.К. Обработка резанием. Управление стабильностью и качеством в автоматизированном производстве / М.: Машиностроение, 1989, с. 296

9. Дубинин П.И. К вопросу классификации обработки природного камня. - М., Горный информационно-аналитический бюллетень, №5,2006, с.36-48

10. Дубинина А.П. Влияние технологических режимов на качество обрабатываемых поверхностей алмазов при групповой огранке. - М., Горный информационно-аналитический бюллетень, №9, 2005, с. 321-325.

11. Дубинина А.П. Метод групповой огранки мелких алмазов. — Сб. науч. тр. IV Международной научно-технической конференции «Добыча, обработка и применение природного камня». - Магнитогорск: МагГТУ, 2004, с.224-227.

12. Дубинина А.П. Особенности технологического процесса групповой огранки алмазов в режиме пластического шлифования. -- Труды VII Всероссийской научной конференции «Дизайн и технология художественной обработки материалов», вып. *8.* - Челябинск: Издательство ЮУрГУ, 2004, с. 49-53.

13. Морозов В.И., Дубинин П.И., Дубинина А.П. Тенденции развития художественной обработки природного камня в России. - Труды XI Международного симпозиума «GEOTECHNIKA - GEOTECNTCS 2004», Польша, г.Усгронь, 2004, с. 51-58.

14. Сильченко О.Б., Дубинина А.П. Возможности применения метода пластических деформаций в мезообъемах для групповой огранки алмазов в бриллианты. - М, Горный информационно-аналитический бюллетень, №4, 2004, с. 275-276.

15. Сильченко О.Б., Дубинина А.П. Критическая технология размерно-регулируемой бездефектной обработки твердоструктурных минералов микро-шлифованием. - Сб. науч. тр. Международной конференции «VI Школа геомеханики», Польша» г. Устронь, 2003, с. 167-180.

16. Сильченко О.Б., Дубинина А.П. Критические технологии обработки сверхтвердых материалов. - М., Горный информационно-аналитический бюллетень, №3, 2004, с. 139-141.

17. Сильченко О.Б., Дубинина А.П. Особенности алмазно-абразивной обработки минералов в режиме пластического шлифования на

станке АН15Ф4. -М., Драгоценные металлы. Драгоценные камни, №6(126), 2004, с. 94-96.

18. Сильченко СБ., Дубинина А.П. Особенности групповой алмазноабразивной обработки минералов в режиме пластического шлифования на стайке АН15Ф4. - Труды XI Международного симпозиума «GEOTECHNIKA -GEOTECIINICS 2004», Польша, г. Устронь, 2004, с. 85-89.

19. Сильченко О.Б., Попов BJL, Дубинина А.П. Повышение производительности и экологичности производства бриллиантов за счет применения группового метода огранки алмазов. - Материалы конференции «VII SZKOLA GE-OMECHANIKI», Польша, Устронь, 2005, с. 493-497.

20. Н. Королев, Д. Королев. AVR-микроконтроллеры второго поколения: средства разработчика // Компоненты и Технологии. 2003, с. 112.

21. Н. Королев, Д. Королев. AVR-микро контроллеры второго поколения: новые аппаратные возможности // Компоненты и Технологии. 2003, с. 13.

22. Н. Королев, Д. Королев. AVR-микро контроллеры: большое в малом // Схемотехника. 2001, с.10.

23. Н. Королев, Д. Королев. AVR-микро контроллеры: программные средства // Компоненты и Технологии. 2000, с.14.

24. Н. Королев. AVR: аппаратные средства разработчика // Компоненты и Технологии. 1999, с.30-32.

25. Глушаков СИ., Сурядный А.С. Программирование на Visual Basic 6.0. - М.: ООО "Издательство АСТ"Ларьков «Фолио», 2003, с. 497

26. Кондрашов Ю.Н. Visual Basic для Windows. Формы и элементы управления. Учебное пособие. - М.: изд. Академия бюджета и казначейства, 1997, с. 242

27. Кондрашов Ю.Н. Visual Basic 6.0. Описание языка. Основные элементы управления: учебное пособие / Ю.Н. Кондрашов, И.А.

Мещерякова, Лебедев В.М. - М., Академия бюджета и казначейства. 2003, с.77

28. Лебедев В.М., Мещерякова Н.А., Распутин А.П. и др. Основные возможности Visual Basic 6.0 для работы с файлами, графикой и базами данных: Учебное пособие. Омск: ООИПКРО, 2004, с.88

29. Мещерякова Н.А. Алгоритмизация и обьектно-ориентированное программирование в курсе информатики высших учебных заведений экономического профиля // Компьютеризация обучения и проблемы гуманизации образования в техническом вузе: материалы Международной научно-методической конференции - Пенза, 2003, с. 301-305

30. Плясунов А. В. 06 одном подходе к решению эадач двухуровневого программирования//Труды XII Байкальской между нар. конф. «Методы оптимизации и их приложении». Иркутск, 2001, с. 227-231.